1974

No-drop bomb simulation using micro-computers.

Pease, Andrew John.

Monterey, California. Naval Postgraduate School

NO-DROP BOMB SIMULATION USING MICRO-COMPUTERS

Andrew John Pease

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NO-DROP BOMB SIMULATION USING

MICRO-COMPUTERS

by

Andrew John Pease

June 1974

Thesis Advisor: U. R. Kodres

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NO-DROP BOMB SIMULATION USING MICRO-COMPUTERS		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis June 1974 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Andrew John Pease		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1974 13. NUMBER OF PAGES 126
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Practice Bombing; Simulation; Micro-Computer; Aviation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The implementation of a no-drop bomb simulation system on a micro-computer is presented. The bomb trajectory algorithms supplied by Mr. Lee Thomson, China Lake Naval Weapons Center, formed a base for the simulation programs.

The simulation program was written in FORTRAN to test the four phases of the system: tracking, position and velocity

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 - ABSTRACT (Cont.)

fitting, trajectory simulation, and impact point calculation. Translated PL/M programs form a base for the micro-computer implementation.

No-Drop Bomb Simulation Using Micro-Computers

by

Andrew John Pease
Ensign, United States Navy
B.S., United States Naval Academy, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1974

ABSTRACT

The implementation of a no-drop bomb simulation system on a micro-computer is presented. The bomb trajectory algorithms supplied by Mr. Lee Thomson, China Lake Naval Weapons Center, formed a base for the simulation programs.

The simulation program was written in FORTRAN to test the four phases of the system: tracking, position and velocity fitting, trajectory simulation, and impact point calculation. Translated PL/M programs form a base for the micro-computer implementation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure

ACKNOWLEDGEMENT

The simulation project was created by Lt. Alfred Pease, USN and is under the direction of Associate Professor Uno Kodres.

The author wishes to express his thanks to Lt. Pease, China Lake Naval Weapon Center, for his assistance in the overall system design, Associate Professor Kodres for his overall help and guidance throughout the project, and Assistant Professor Gary Kildall for his assistance in the microcomputer programming. Special thanks go out to my friends Donna Sadel and Mike Harris who greatly assisted in maintaining the author's sanity throughout the preparation of this project. A special thanks is also in order to my parents who have constantly served as an inspiration throughout this thesis project.

I. INTRODUCTION

The No-Drop Bomb Simulation on a micro-computer, consists of a series of algorithms which are programmed to simulate the fall of a bomb in a practice environment. The advantages of such a system are numerous, both in cost and flexibility.

Presently, the Department of the Navy spends in excess of $1 million for the purchase of practice bombs for pilot training.[1] These bombs require storage space on the aircraft carriers for at sea training. Since these practice bombs are live ordnance, there are many restrictions placed both on practice ranges and training flight plans.

A simulation of the bomb drop will eliminate the need for these practice bombs, in exchange for a simulation micro-computer interfaced with a radar, self-contained in a portable van. The van will then be positioned near a designated target location prior to the practice bombings. Since there is no live ordnance attached to the aircraft, the targets themselves can be more realistic; for example bridges, buildings or industrial sites. The pilot's primary restriction in number of runs will be the amount of fuel he carries. He may simulate bombing runs as often as the training session

---

[1] Based on oral communication with the Naval Test and Evaluation Squadron (VX-5), China Lake, Calif.

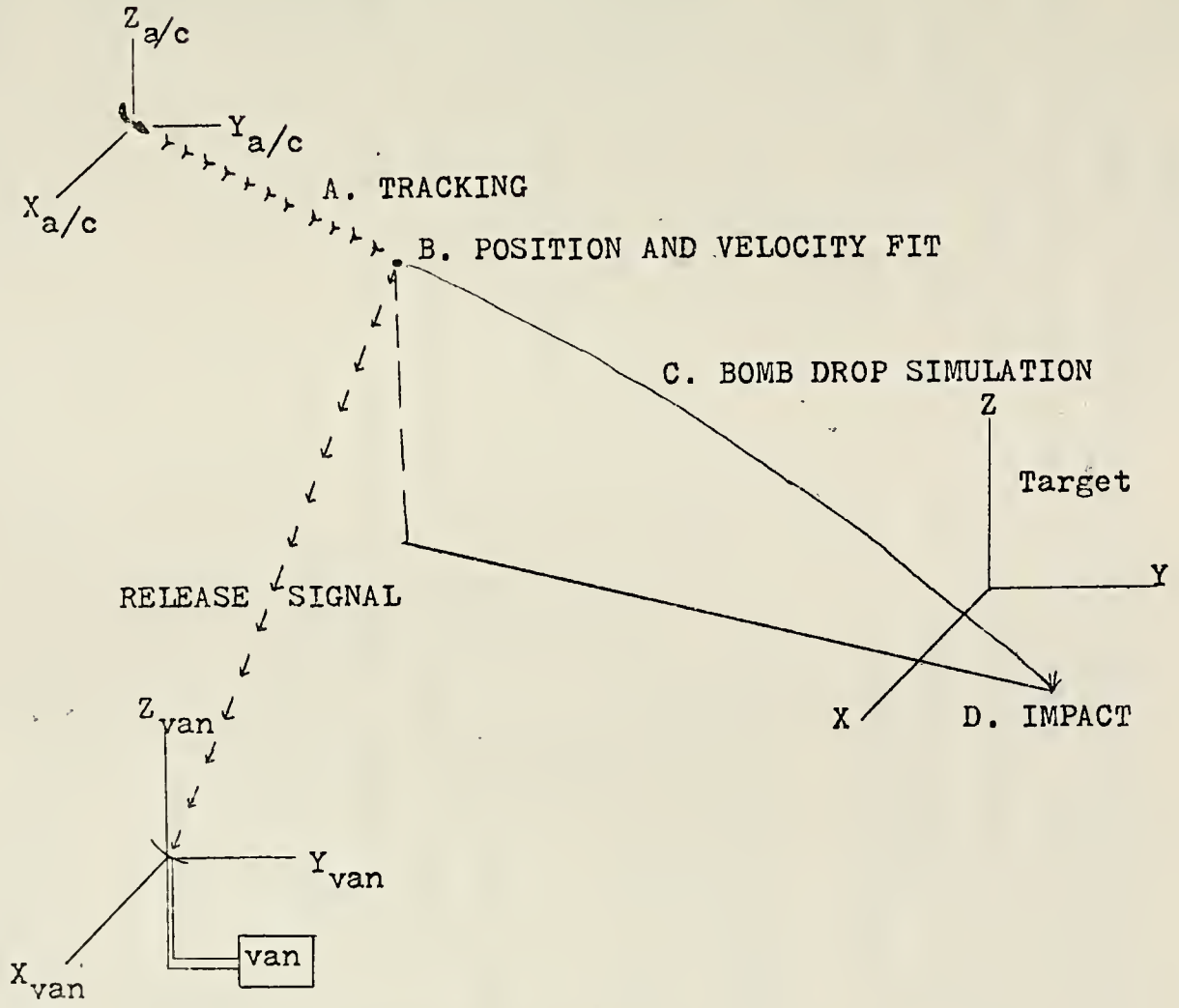
dictates with a variety of bombs. Training session time is increased due to the elimination of bomb loadings and mechanical failures.

The biggest advantage, however, exists in the actual training of the pilot. There are only three pilot controlled factors which determine a target hit or miss: dive angle, velocity, and altitude. With practice ordnance on board the aircraft, an uncontrollable variable is introduced-ballistic dispersion. The random ballistic dispersion serves only to confuse the pilot. Therefore, the release conditions should be the only variables that determine the practice run outcome.

The simulation process is subdivided into four phases: the tracking of the aircraft, determining the aircraft's position and release velocity, simulating the bomb fall, and calculating the impact point. (See Figure 1).

A Nike Hercules radar will continuously track the aircraft in a spherical coordinate system. An analog to digital converter will provide digital input to the micro-computer. The micro-computer will store the current position history of the aircraft in three arrays, each containing the sixteen most current spherical coordinate values. (Section II-A).

When the pilot releases the bomb, a tone signal will be sent to the micro-computer. A least squares polynomial fit using the current aircraft position history yields the spherical coordinate release position and velocity. This position and velocity is translated into a Cartesian coordinate system. (Section II-B).



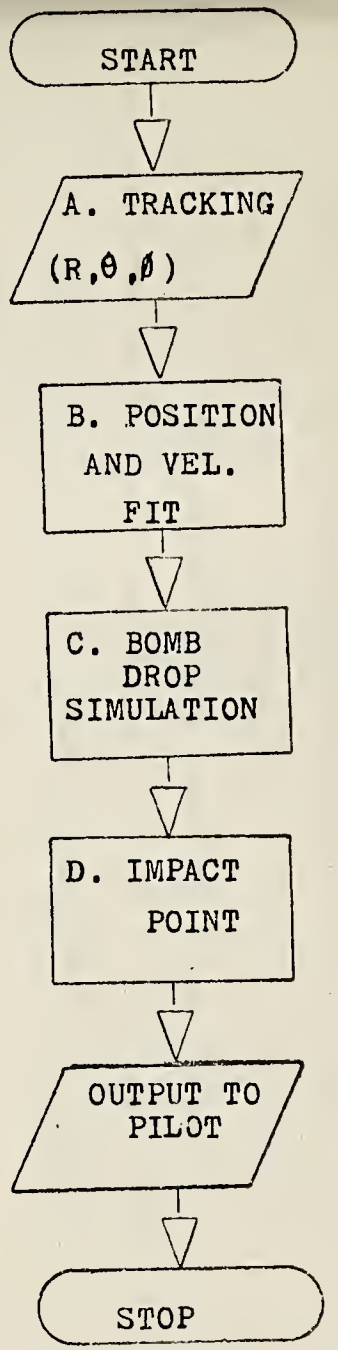


Figure 1. The Four Phases of the Simulation Process

Using the calculated release conditions, the bomb drop trajectory is assumed to be governed by a system of differential equations which are solved numerically using the Runge-Kutta fourth order integration scheme. The bomb altitude is continuously compared with the target's altitude for impact. (Section II-C).

The impact coordinates are finally calculated using the bomb's down range travel and time of fall. The velocity of the wind is taken into consideration in deriving two sets of coordinates for the impact point, one with respect to the target's fixed run-in-line and the other with respect to the aircraft's run-in-line. (Section II-D).

The simulation was initially programmed using FORTRAN for ease of testing and debugging. Important phases in programming were in calculating the aircraft's release conditions and simulating the bomb drop trajectory.

The trajectory phase of the process was tested independently for accuracy in results. This made it possible to use known release conditions for an output comparison with published results. The routine was subsequently optimized for the micro-computer implementation. (Section II-A). The release condition curve fitting process was similarly tested using actual recorded data as the standard.

Upon completion of the FORTRAN implementation, the resultant program was translated into PL/M, a high level language compatible with the selected micro-computer (INTELLEC 8). Because suitable arithmetic operations were not

available, systems implementation involved considerable additional effort in the development of a floating point package for the arithmetic operations. (Section IV-A).

Due to time limitations, the project came to an end with all the PL/M routines compiled but untested.

II. BALLASTIC PROBLEM

The ballastics problem is subdivided into four distinct phases: tracking, position and velocity fitting, bomb trajectory simulation, and deriving the impact point. A radar tracking system provides the micro-computer with a digital spherical coordinate history of the aircraft's position at a predetermined time interval. At the bomb's release point, a least squares data fit allows the calculation of the Cartesian position and velocity components at release. The bomb's trajectory towards the target is then calculated yielding a down range travel and time of fall of the bomb. The impact point is then derived with respect to the fixed target's run-in-line and the aircraft's run-in-line. The initial release conditions and the impact point are then relayed to the pilot.

A. TRACKING PHASE

The initial phase of the overall simulation involves tracking the aircraft until the pilot releases the bomb. The tracking phase samples the radar's special spherical coordinate inputs at a fixed time interval yielding R, θ, and ϕ.[2] (See Figure 2).

These input values are stored in three wrap-around arrays corresponding to R, θ, and ϕ. In order to minimize the

---

[2] Special coordinate system based on the radar's output.

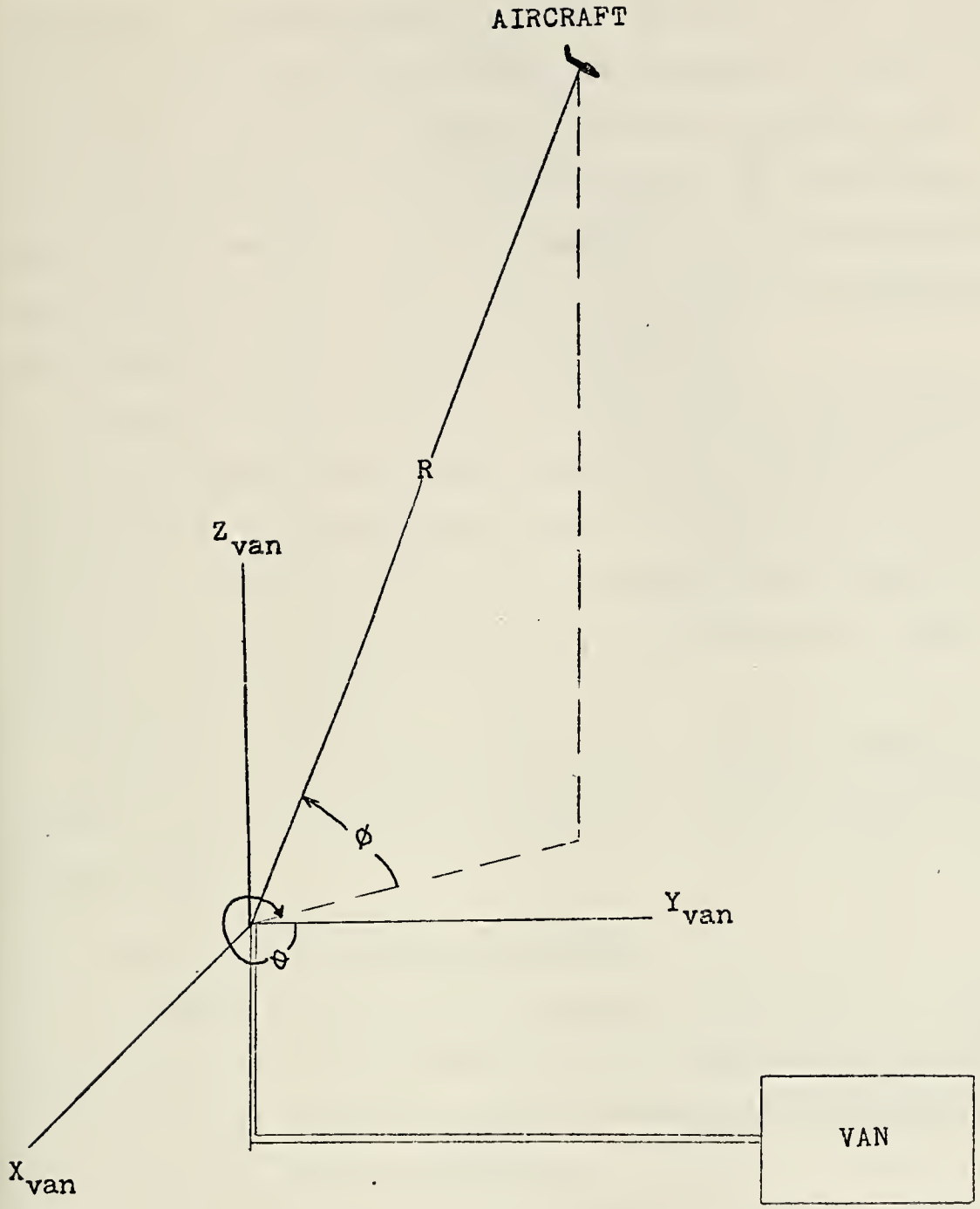


Figure 2. Radar Tracking System Coordinates

arithmetic processing and storage, the values stored in these arrays are the actual radar outputs subtracted from the base value. Since each of the arrays are cyclical in nature, new inputs are read over old input values thus retaining the most recent sixteen point position history of the aircraft. Each array consists of sixteen elements and two associated base values: Tilda and Bar. Two base values are needed for the wrap-around type input. (See Figure 3).

When the table is full, a new base value is calculated and the old base value is set equal to the current base value ($\bar{R} \leftarrow \tilde{R}$). The current base value is set equal to the current radar output value ($\tilde{R} \leftarrow R$), and the value stored in the table is equal to the current base value minus the current radar output value ($\tilde{\Delta} \leftarrow \tilde{R} - R$).

Once the pilot releases the bomb, a current spherical coordinate position history is available with the associated base values.

B. POSITION AND VELOCITY FIT PHASE

To reduce the number of arithmetic operations, the least squares position and velocity fit was performed on the spherical coordinates and subsequently translated to Cartesian position and velocity components.

The least squares fit was performed using the second degree Legendre Discrete Orthogonal Polynomials. The second degree polynomial was used to estimate the position and velocity of the aircraft at the release point. The Legendre

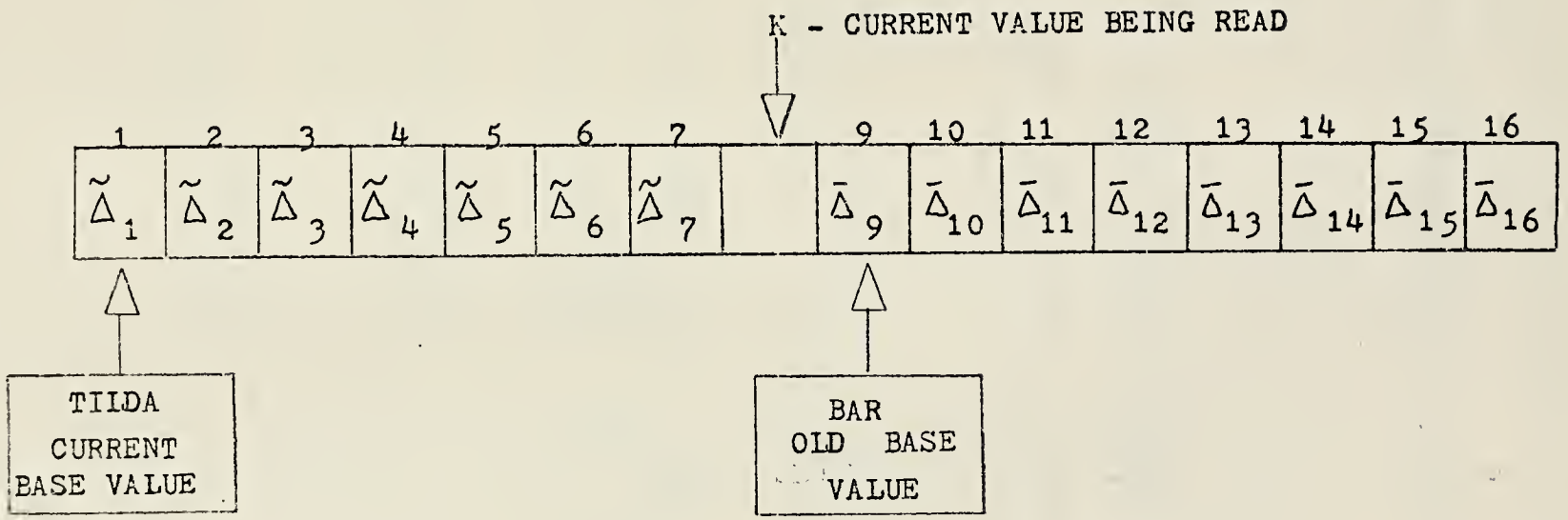


Figure 3.. Wrap-Around Position History Array

Orthogonal polynomials of the second degree were used for a sixteen point fit.

The orthogonal polynomials have the form:

$$O_{0,15}(x) = 1$$

$$O_{1,15}(x) = 1 - 2 * \frac{x}{15}$$

$$O_{2,15}(x) = 1 - 6 * \frac{x}{15} + 6 * \frac{x}{15} * \frac{(x-1)}{14}$$

$$= \frac{14*15 - 6*14*x + 6*x(x-1)}{14*15}$$

The values of these polynomials are based on the number of points to be fit. (See Table I for polynomial tabulations).

The second degree polynomial which fits the data is given by the formula below. For further reference see McCalla [1].

$$F(x) = a_0 + a_1(1 - 2*\frac{x}{15}) + a_2(1 - 6*\frac{x}{15} + 6*\frac{x}{15}*\frac{(x-1)}{14})$$

$$= (a_0 + a_1 + a_2) - (\frac{2*a_1}{15} + \frac{6*a_2}{15} + \frac{6*a_2}{14*15})*x + \frac{6*a_2}{14*15}*x^2$$

where:

$$a_0 = \frac{\sum_{I=1}^{16} O_{0,(I-1)} * \text{Track (I)}}{\sum_{I=1}^{16} O_{0,(I-1)}^2}$$

$$a_1 = \frac{\sum_{I=1}^{16} O_{1,(I-1)} * \text{Track (I)}}{\sum_{I=1}^{16} O_{1,(I-1)}^2}$$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$O_{0,x}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$O_{1,x}$	1	$\frac{13}{15}$	$\frac{11}{15}$	$\frac{9}{15}$	$\frac{7}{15}$	$\frac{5}{15}$	$\frac{3}{15}$	$\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{3}{15}$	$-\frac{5}{15}$	$-\frac{7}{15}$	$-\frac{9}{15}$	$-\frac{11}{15}$	$-\frac{13}{15}$	-1
$O_{2,x}$	1	$\frac{126}{210}$	$\frac{54}{210}$	$-\frac{6}{210}$	$-\frac{54}{210}$	$-\frac{90}{210}$	$-\frac{114}{210}$	$-\frac{126}{210}$	$-\frac{126}{210}$	$-\frac{114}{210}$	$-\frac{90}{210}$	$-\frac{54}{210}$	$-\frac{6}{210}$	$\frac{54}{210}$	$\frac{126}{210}$	1

Table I. SECOND DEGREE LEGENDRE DISCRETE POLYNOMIALS EVALUATED FOR A 16 POINT FIT

$$a_2 = \frac{\sum_{I=1}^{16} O_{2,(I-1)} * \text{Track}(I)}{\sum_{I=1}^{16} O_{2,(I-1)}^2}$$

where Track (I) refers to one of the data arrays R, θ, or ϕ.

For example, if release occurs at position K in the track array, $O_{0,0}$ is multiplied by Track(K+1). Since the track array values are differences, the proper base value must be applied. Therefore, for release at point K, the a_j's ($j \leftarrow 0,1,2$) are calculated as follows:

$$a_j = \frac{\sum_{I=1}^{(16-K)} O_{j,(I-1)} * [\text{Track}(K+I)+\text{Bar}] + \sum_{I=K+I}^{16} O_{j,(I-1)} * [\text{Track}(I-K)+\text{Tilda}]}{\sum_{I=1}^{16} O_{j,(I-1)}^2}$$

however $\sum_{I=1}^{16} O_{j,(I-1)}^2$, Bar, and Tilda are constants with respect to the summation, therefore:

$$a_j = \frac{1}{\sum_{I=1}^{16} O_{j,(I-1)}^2} * \left[\sum_{I=1}^{(16-K)} O_{j,(I-1)} * \text{Track}(K+I) + \text{Bar} * \sum_{I=1}^{(16-K)} O_{j,(I-1)} \right.$$

$$\left. \text{Tilda} * \sum_{I=K+1}^{16} O_{j,(I-1)} + \sum_{I=K+1}^{16} O_{j,(I-1)} * \text{Track}(I-K) \right]$$

It can be seen that the sum of the products using the Track differences minimizes the arithmetic processing.

To evaluate F(x) at x=15 (release time) we obtain:

$$F(15) = (a_0 + a_1 + a_2) - \left(2*a_1 + 6*a_2 + \frac{6*a_2}{14}\right) + \frac{6*a_2}{14} * 15$$

$$= (a_0 - a_1 + a_2) \quad \text{Smoother position at release.}$$

To evaluate $F'(x)$ at $x=15$ we obtain:

$$F'(15) = -\frac{1}{15} * (2*a_1 + 6*a_2 + \frac{6*a_2}{14}) + \frac{12}{14} * a_2$$

$$= -\frac{2}{15} * a_1 + \frac{6}{14} * a_2 \quad \text{Velocity at release.}$$

The special spherical coordinate positions and velocities are transformed to the Cartesian coordinate positions and velocities using the following formulas: (See Figure 4).

$$X_v = R * \text{Sin } \theta * \text{Cos } \phi$$

$$Y_v = R * \text{Cos } \theta * \text{Cos } \phi$$

$$Z_v = R * \text{Sin } \phi$$

thus taking time derivatives:

$$\dot{X}_v = \text{Sin } \theta * \text{Cos } \phi * \dot{R} + R * \text{Cos } \theta * \text{Cos } \phi * \dot{\theta} + R * \text{Sin } \theta * \text{Sin } \phi * \dot{\phi}$$

$$\dot{Y}_v = \text{Cos } \theta * \text{Cos } \phi * \dot{R} + R * \text{Sin } \theta * \text{Cos } \phi * \dot{\theta} - R * \text{Cos } \theta * \text{Sin } \phi * \dot{\phi}$$

$$\dot{Z}_v = \text{Sin } \theta * \dot{R} + R * \text{Cos } \theta * \dot{\theta}$$

The reader will note that the X_v, Y_v, and Z_v position coordinates are with respect to the radar's (van) axis. The positions are translated by using the van's displacement coordinates from the target center (X_{van}, Y_{van}, and Z_{van}) to put position coordinates in the target's frame of reference.

$$X = X_v + X_{van}$$

$$Y = Y_v + Y_{van}$$

$$Z = Z_v + Z_{van}$$

(See Figure 5).

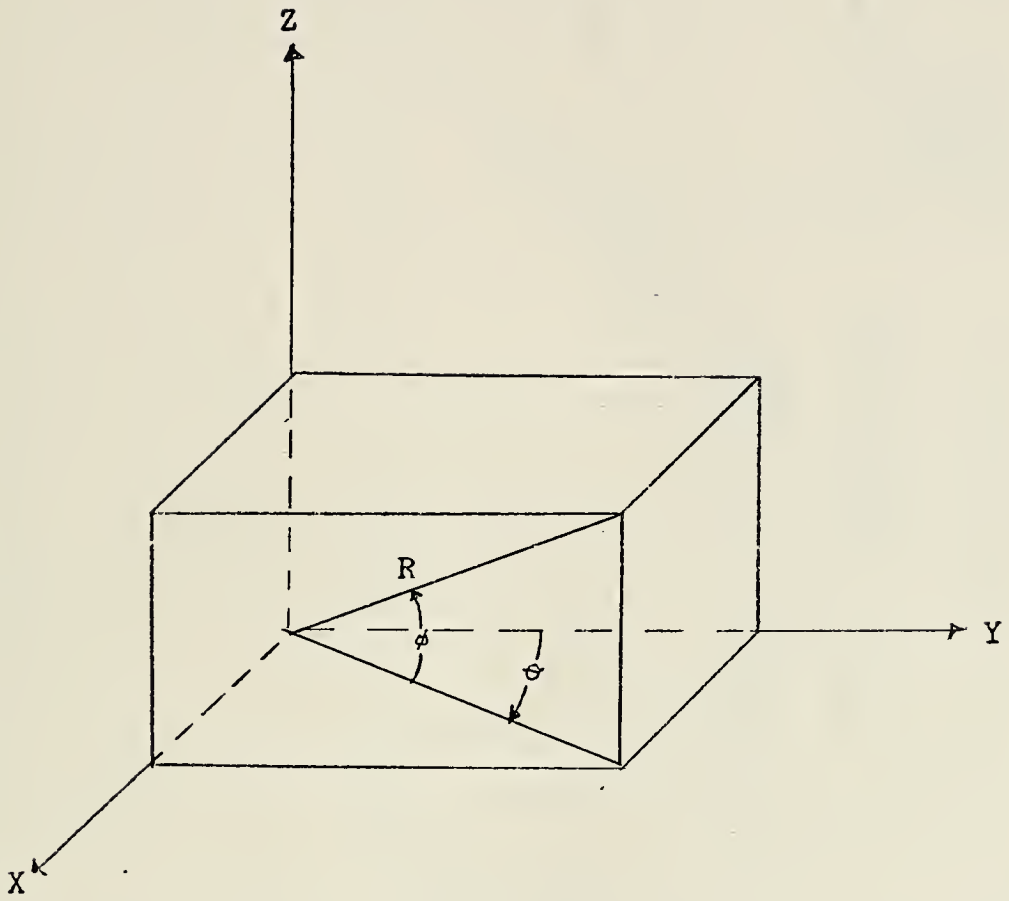
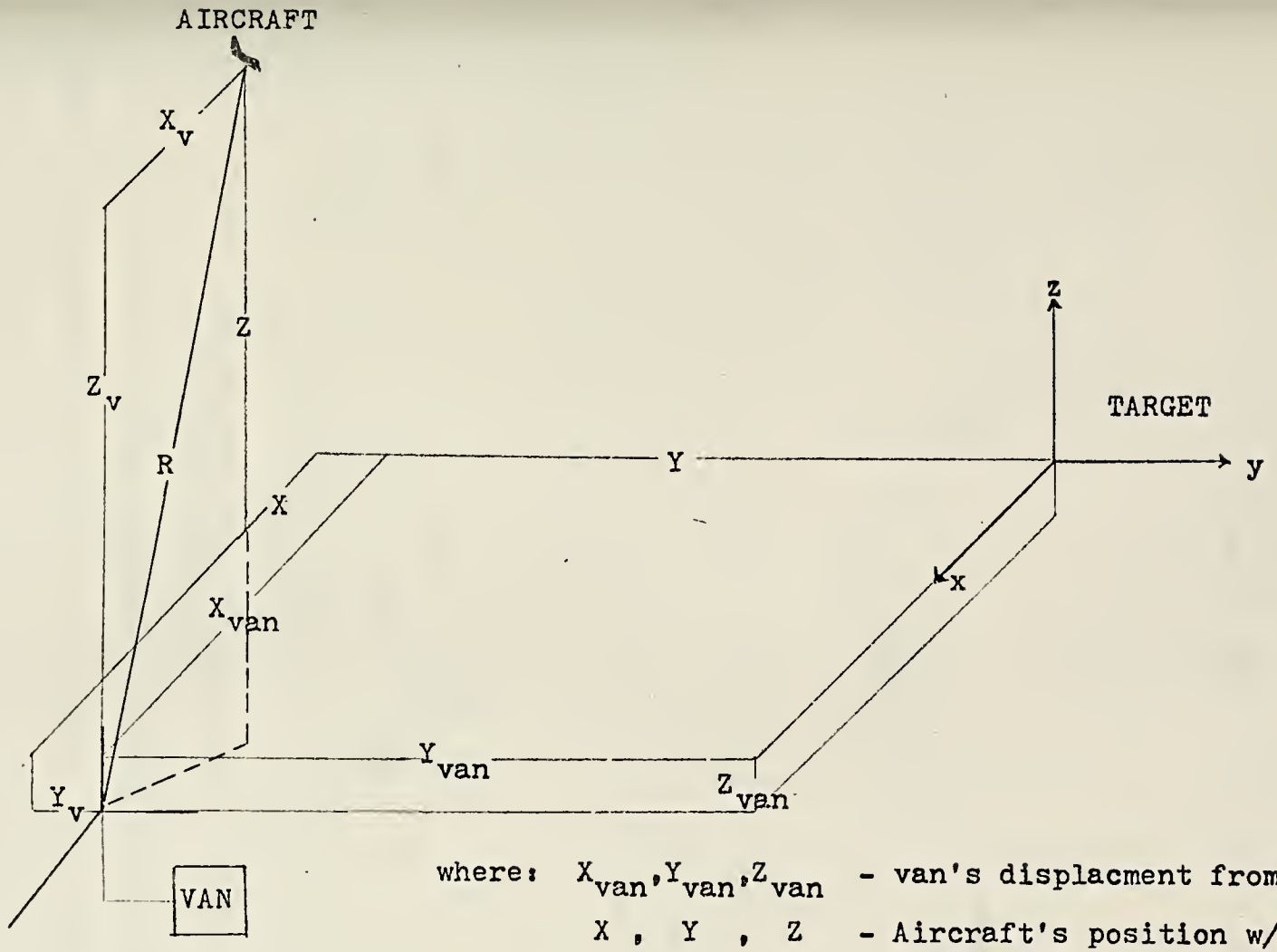


Figure 4. Transformation of Radar Tracking Coordinates into Cartesian Coordinates.



where: $X_{van}, Y_{van}, Z_{van}$ - van's displacment from target center

X , Y , Z - Aircraft's position w/ respect to target

X_v , Y_v , Z_v - Aircraft's position w/ respect to van

Figure 5. Coordinate Frames

C. BOMB DROP SIMULATION PHASE

Prior to simulating the bomb drop, the pilot's release conditions, altitude, dive angle, and velocity, must be determined.

Altitude (H) = Z_v

$\underline{V} = (\dot{X}_v, \dot{Y}_v, \dot{Z}_v)$

$Vel = \sqrt{\dot{X}_v^2 + \dot{Y}_v^2 + \dot{Z}_v^2}$

Dive Angle (α) = $ArcTan \left(\frac{\dot{Z}_v}{\sqrt{\dot{X}_v^2 + \dot{Y}_v^2}}\right)$

(See Figure 6).

At this point, the initial velocity and drop angle of the bomb are determined taking into account the ejection velocities (components perpendicular and parallel to the flight path) of the bomb rack canisters.

Thus:

Drop Angle (α) = + $ArcTan \left(\frac{VE3B}{Vel+VE2B}\right)$

Velocity (Vel) = $\sqrt{(Vel + VE2B)^2 + VE3B^2}$

where: VE2B - the ejection velocity parallel to the aircraft's flight path.

VE3B - the ejection velocity perpendicular to the aircraft's flight path.

In the case of a dual drag bomb (a projectile whose aerodynamic characteristics change in flight), fin opening times must be considered. The purpose of a fin opening time is to determine when the bomb's fins open, thus altering the drag characteristics of the projectile.

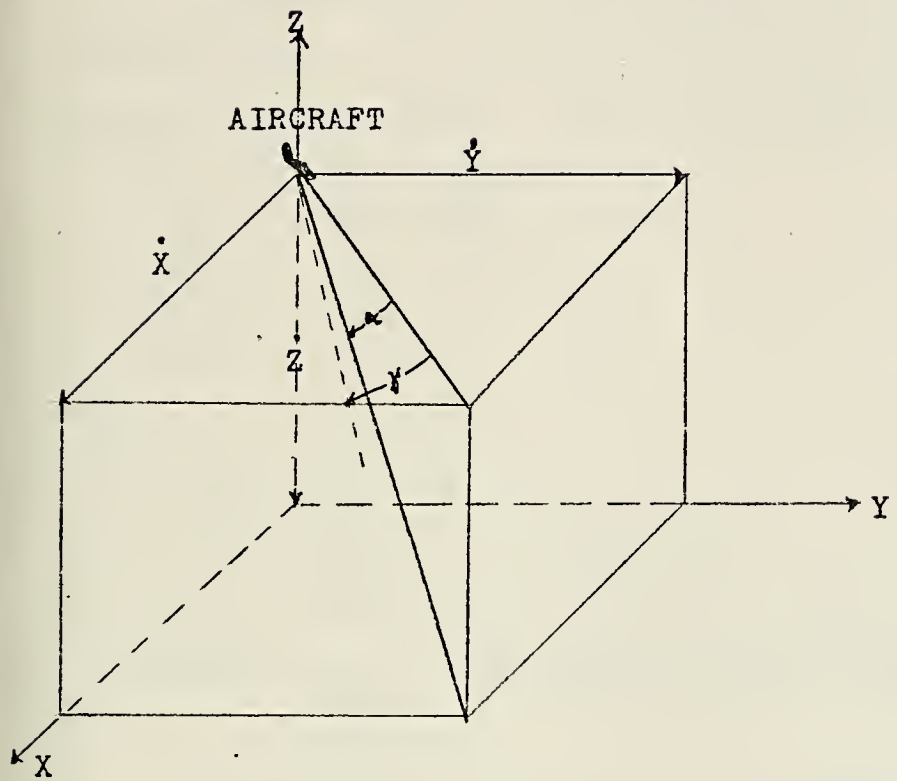


Figure 6. Aircraft's Release Angle α and
Bomb's Release Angle γ

In the bomb drop simulation, it is assumed that the bomb falls straight down the aircraft's run-in-line. Since the bomb has no independent propulsion units and ballastic dispersion is not introduced in the theoretical model, this is a valid assumption.

1. Differential Equations

The mathematical model can be described in vector form as follows. Let x and y be the horizontal and vertical components respectively. From Newton's Equation of Motion:

$$m \frac{d}{dt} V_t = -gm - \frac{\|V_t\|^2}{\|V_t\|} (V_t * C_b)$$

$$m \frac{d}{dt} V_t = -gm - \|V_t\| (V_t * C_b)$$

where:

V_t - total velocity

m - mass of the projectile

g - force due to the earth's gravitation

$(V_t * C_b)$ - sum of the forces acting on the body, where drag is proportional to the square of the velocity

The drag coefficient (C_b) is computed for each type of bomb and it depends on the speed at which the projectile is travelling, the cross-sectional area of the bomb, and the air mass density at a given altitude.

In the component form, the above equation becomes:

$$m \frac{dV_x}{dt} = \|V_t\| (V_x * C_b)$$

$$m \frac{dV_y}{dt} = -gm - \|V_t\| (V_y * C_b)$$

$$\frac{dx}{dt} = V_x$$

$$\frac{dy}{dt} = V_y$$

where x and y represent horizontal and vertical displacements respectively, and where V_x and V_y are horizontal and vertical velocity components.

a. Air Density Computation

Given the hydrostatic equation $\frac{dp}{dz} = -\rho g$ (1)

and the equation of state $p = \rho RT$ (2)

where: p = pressure

ρ = density of air

z = altitude

g = gravitational force (32 ft/sec^2)

T = temperature

R = gas constant

Solving equation 2 for ρ and substituting into equation 1:

$$\frac{dp}{dz} = -\frac{p}{RT} g$$

$$\frac{1}{p}\frac{dp}{dz} = -\frac{g}{RT}$$

$$\frac{d(\ln p)}{z} = -\frac{g}{RT}$$

Take the integral thru some altitude Z:

$$\int_{z=0}^{Z} \frac{d(\ln p)}{dz}\, dz = -\int_{z=0}^{Z} \frac{g}{RT}\, dz$$

Assume T is constant.

$$\ln p(z) - \ln p(0) = -\frac{gz}{RT}$$

$$\ln\left(\frac{p(z)}{p(0)}\right) = -\frac{gz}{RT}$$

$$p(z) = p(0)\, e^{-\frac{gz}{RT}}$$

From equation 2, $\rho = \frac{p}{RT}$

$$\rho = \frac{p_0}{RT}\, e^{-\frac{gz}{RT}}$$

Assuming g/RT to be constant, we reduce to

$$\rho = \rho_0\, e^{-\frac{z}{H}}$$

where: $H = \frac{RT}{g}$

ρ_0 = density at z=0.

Thus when z = H,

$$\rho(H) = \rho(0)\, e^{-1}$$

$$\rho(H) = \frac{\rho(0)}{e}$$

$$\frac{\rho(H)}{\rho(0)} = \frac{1}{e}$$

Thus, we must choose H such that

$$\frac{\rho(H)}{\rho(0)} = \frac{1}{e}$$

$\rho(0) = 0.00237691$ slugs/ft^3 (Density of air at sea level.)

Solving for e,

$\rho(H) = 0.0008744$ slugs/ft^3

Solving for H from a fourth order density equation,

H = 30500 ft.

Therefore,

$$\rho(z) = \rho_0\, e^{-\frac{z}{30500}}$$

Using this formula, a divided difference density table was computed with the altitude as the independent variable.

Similar divided difference tables were constructed for the speed of sound based on altitude and the drag coefficient based on the Mach number of the aircraft.

2. The Integration Scheme

The system of differential equations was solved by a standard fourth order Runge-Kutta integration scheme which was selected for its computational efficiency. Let the four differential equations in the system be represented by:

$$\frac{du_i}{dt} = f_i(u_1,u_2,u_3,u_4) \quad i = 1,2,3,4$$

The application of the Runge-Kutta scheme gives, for i=1,2,3,4

$$m_{1i} = \Delta t * f_i(u_1,u_2,u_3,u_4)$$

$$m_{2i} = \Delta t * f_i(u_1+\tfrac{1}{2}m_{11},u_2+\tfrac{1}{2}m_{12},u_3+\tfrac{1}{2}m_{13},u_4+\tfrac{1}{2}m_{14})$$

$$m_{3i} = \Delta t * f_i(u_1+\tfrac{1}{2}m_{21},u_2+\tfrac{1}{2}m_{22},u_3+\tfrac{1}{2}m_{23},u_4+\tfrac{1}{2}m_{24})$$

$$m_{4i} = \Delta t * f_i(u_1+m_{31},u_2+m_{32},u_3+m_{33},u_4+m_{34})$$

$$u_i' = u_i + \frac{1}{6}(m_{1i}+\tfrac{1}{2}m_{2i}+\tfrac{1}{2}m_{3i}+m_{4i})$$

$$\text{time} = \text{time} + \Delta t$$

The variables are identified as follows:

$u_1 = V_x$ $\qquad$ $f_1 = \frac{1}{m}\|V_t\|\ (V_x * C_b)$

$u_2 = y$ $\qquad$ $f_2 = dy/dt$

$u_3 = x$ $\qquad$ $f_3 = dx/dt$

$u_4 = V_y$ $\qquad$ $f_4 = \frac{1}{m}\|V_t\|\ (V_y * C_b) - g$

The trajectory simulation utilizes this integration process to time step the projectile towards the target. After every Δt incrementation, the altitude (y position) is compared with the target altitude. If the simulated bomb is above target altitude, the integration is refreshed and updated. The present position and velocity components are saved for possible interpolation. If, because of a large time interval, the bomb travels past target altitude, a linear interpolation is used to reduce the time interval. The previously saved position and velocity components are restored and the integration is performed using the interpolated Δt. Impact occurs when the projectile altitude is within 1.2 inches of target altitude. At impact time, the values x, y, V_x, and V_y are retained for the next phase.

D. IMPACT PHASE

The impact point must be computed in two frames of reference in order to be of benefit to the pilot. It must be calculated with respect to the target's run-in-line and the aircraft's run-in-line.

Thus two run-in-line angles must be computed, an uncorrected angle, $\gamma_1 = \text{ArcTan } (\frac{\dot{X}}{\dot{Y}})$ for the aircraft's impact point, and a corrected γ due to the velocity component of the effective wind EW = (Xwind, Ywind)

$$\gamma = \text{ArcTan } (\frac{\dot{X} + \text{Xwind}}{\dot{Y} + \text{Ywind}}) \text{ for the target's impact points.}$$

(See Figure 7).

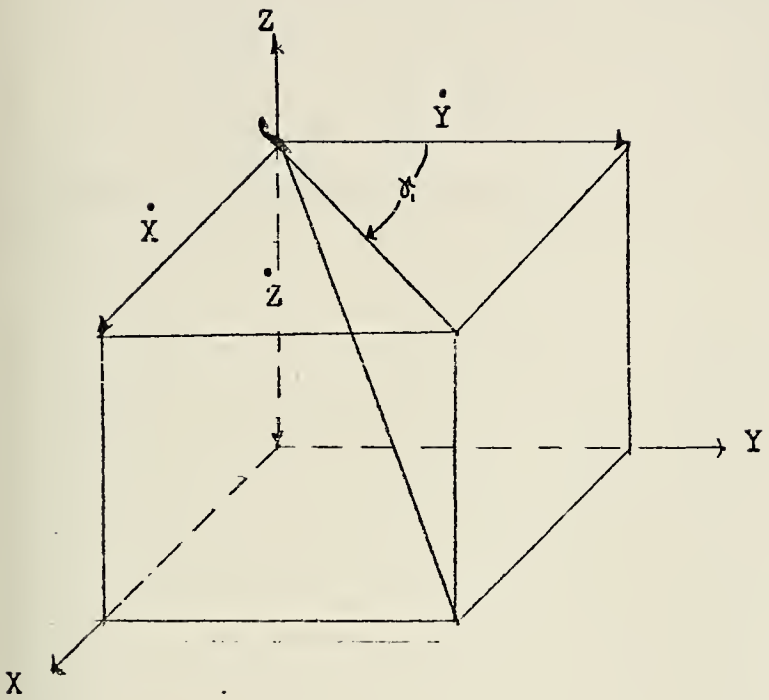


Figure 7. Aircraft's Run-In-Line Angle γ_i

Using DR as the down range travel of the bomb and Time as the total time for the bomb to reach the impact point (See Figure 8), the impact points are calculated as follows:

1) the target's impact coordinates,

$$I_x = DR * \text{Sin } \gamma - X_{a/c} - Xwind*Time$$

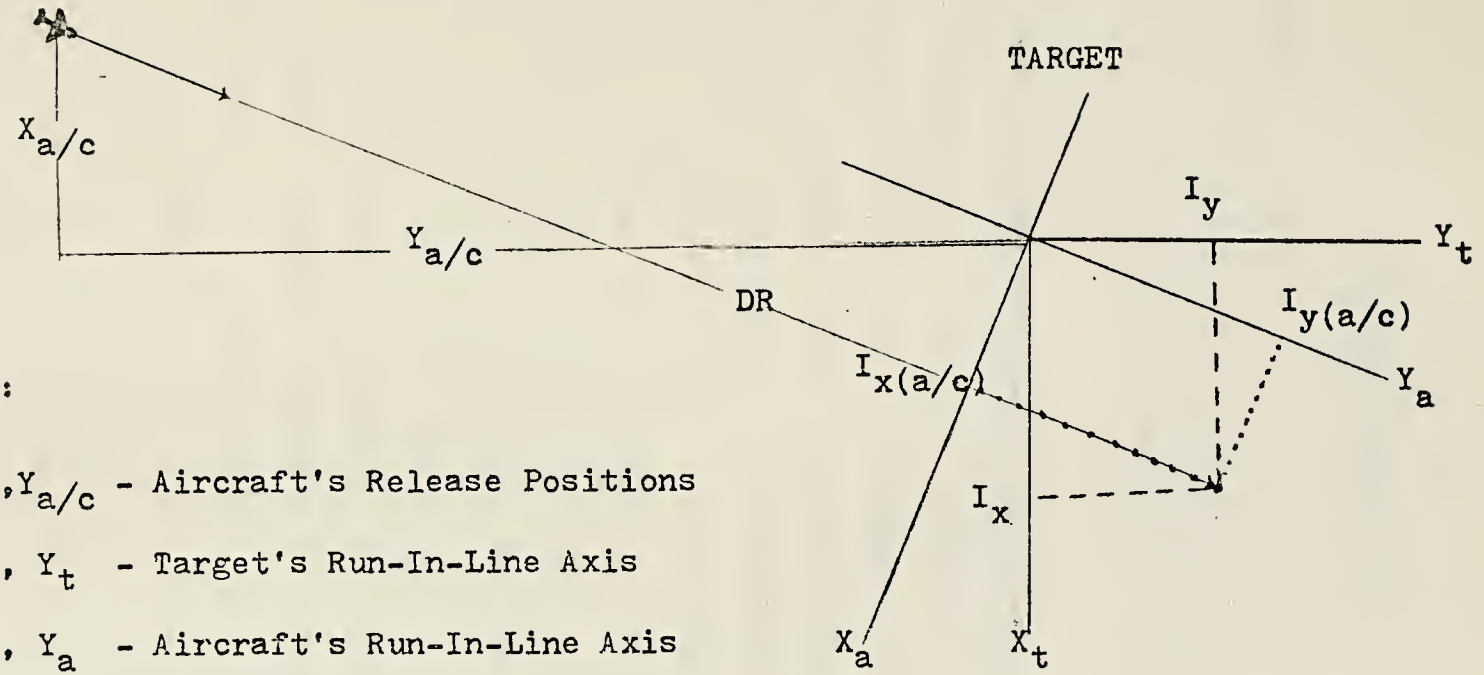
$$I_y = DR * \text{Cos } \gamma - Y_{a/c} - Ywind*Time$$

2) the aircraft's impact coordinates,

$$I_{x(a/c)} = I_x * \text{Cos } \gamma_1 - I_y * \text{Sin } \gamma_1$$

$$I_{y(a/c)} = I_x * \text{Cos } \gamma_1 - I_y * \text{Cos } \gamma_1$$

where: DR - down range travel of the bomb.



WHERE:

$X_{a/c}, Y_{a/c}$ - Aircraft's Release Positions

X_t , Y_t - Target's Run-In-Line Axis

X_a , Y_a - Aircraft's Run-In-Line Axis

I_x , I_y - Target's Impact Coordinates

$I_{x(a/c)}, I_{y(a/c)}$ - Aircraft's Impact Coordinates

Figure 8. Impact Point Calculations

III. FORTRAN SIMULATION

A. BOMB DROP SIMULATION

The initial algorithms for simulating the TRAJ routine were supplied by Lee Thomson of the Naval Weapons Center, China Lake, California.

Euler's integration scheme was used in updating velocity and position components. The time interval set set at .01 seconds for accuracy and thus the number of calls made to the integration routine (EINT) was of the order of 10^3. Due to processing time restrictions, the Runge-Kutta method was selected as an integration scheme. The number of calls to EINT were hence reduced by a magnitude of one hundred, with results differing in the fifth decimal place.

The TRAJ routine was tested using several release conditions to demonstrate the model flexibility in simulating bomb loft (nuclear weapons), level flight release, and dive bombing conditions. Results compared favorably to published results with mean error rates of 1.15% for time of fall and .74% for down range travel.

Choosing a proper time interval for the Runge-Kutta method was conducted by varying the time interval from one to five seconds. The emphasis, at this point, was the number of calls to EINT for the projectile to reach target altitude. The results by altering Δt, showed changes in the sixth decimal place, and a substantial decrease in the number of calls to EINT. However, beyond a time interval of three seconds,

there was only a slight decrease in calls. This was due to the time interval interpolation necessary for the projectile to reach the target. The savings made in reduction of integration calls were offset by increases in the interpolation section. The results of these test runs appear in Appendix A.

B. OVERALL SIMULATION

Divisions are the most time-consuming operations on the micro-computer. The number of divisions in the overall system was therefore greatly reduced by replacing constants used in the division operations by their reciprocals. This was done to replace the division process by the less time consuming multiplication process.

The divided difference table look-up routine was implemented to save in the number of arithmetic operations. The routine is used to obtain values for the air mass density, the speed of sound, and the drag coefficients necessary in the TRAJ routine. The initial call to TABLE searches the table returning the proper value, and stroes the position in the table where the value was found. Subsequently, this position is used as the first estimate for table location on the next call for corresponding entries. Because the values of independent variables change by a small amount at each iteration, the desired table entry is in the immediate vicinity of the last look-up. This method of computing functions trades execution time with memory space and allows the total calculation to be carried out in a matter of seconds of computer time.

IV. PL/M IMPLEMENTATION

To implement the model on the micro-computer, a floating point arithmetic package was written to maintain a four hexadecimal digit accuracy during calculations. The floating point variable therefore occupies three, eight bit words; two words for the variable mantissa and one word for the mantissa sign bit and the variable exponent. The mantissa of the variable is viewed as a normalized binary number with the left most digit always one. (See Figure 9).

In order to manipulate these variables, an arithmetic subroutine library had to be built. At present, there exist routines that add and subtract, multiply, divide and take the square roots of these floating point variables.

The add routine compares the exponents of the inputs. After performing the proper mantissa shift, making the variable exponents equal, a full sixteen bit add operation is done with adjusted mantissa. To subtract, the mantissa sign bit is switched.

The multiply routine divides each of the input mantissas into four hexadecimal digits. The hex digits are then appropriately multiplied using a table lock-up scheme for the hex product. The result is an eight hex digit number of which only the four most significant digits are used for the product mantissa. The fifth digit is used for rounding. The exponents are then added and the proper sign of the product is masked in the result.

PL/M FLOATING POINT NUMBER

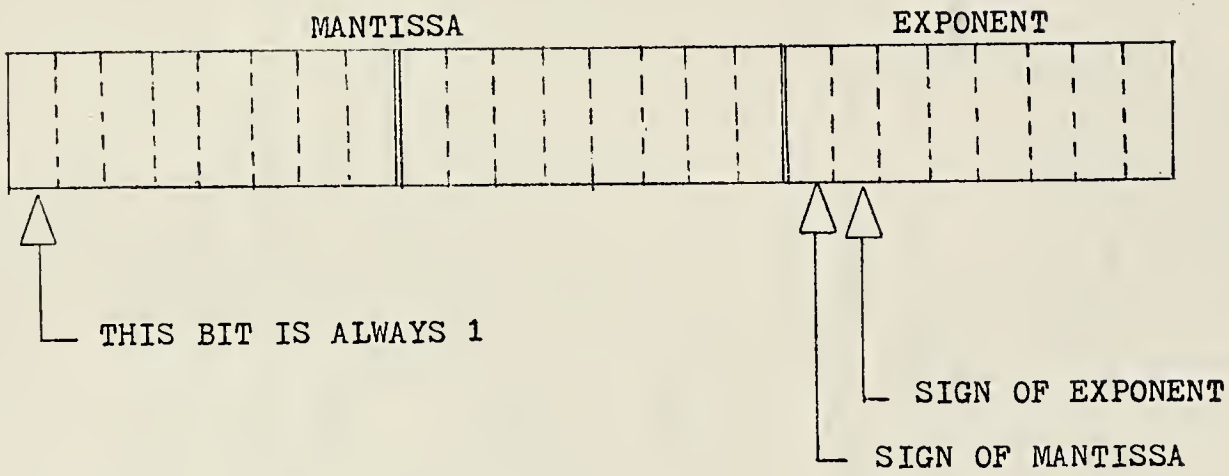


Figure 9.

The divide routine performs a full sixteen bit subtract on the input's mantissa. If the carry bit resulting from the subtraction is on, one is added to the quotient mantissa. The quotient mantissa is then left-shifted, one bit. This procedure is repeated sixteen times after which the original input exponents are subtracted and the proper sign is inserted into the result.

The square root routine utilizes Newton's Iteration for approximating square roots. At present, interrupts have not been implemented for negative inputs.

The actual PL/M version of the simulation program was translated directly from the working FORTRAN version. All associated flow charts, process graphs, program listings, and sample output can be found in the appendix.

The PL/M routines have been compiled, however due to time constraints, none have been tested. Routines which still must be written for full system implementation are Sine, Cosine, and Arctangent trigonometric routines, and input/output procedures. The I/O procedures are important in that they will act as the operator interface with the micro-computer during the actual simulation.

All associated hardware interfaces are in building stages at the Naval Weapons Center in China Lake under the direction of Lt. Alfred Pease.

V. CONCLUSIONS

The purpose of the thesis was to help develop a working system for practice bomb drop simulation and to demonstrate that micro-computers can carry out significant computational tasks. The reader will note that the hardware cost of the MCS 8 micro-computer to be used in the target system would be less than $1000.

Because of the decreasing hardware costs of the micro-computer and significant increases in the processing speeds which the micro-computer industry is presently realizing, new systems which are equivalent to presently available avionics systems can be developed at a hardware cost which is nearly one hundred times smaller. The software development costs are also likely to be smaller because the system can be subdivided into smaller, independently operating subsystems, each of which perform a well defined task.

As an example of future expansion, consider the No-Drop Bomb Simulation in conjunction with a release prediction system. Two micro-computers would be interfaced making it a multiprocessing system. One micro-computer would track the aircraft to the release position while the other micro-computer would perform a drop simulation at the various tracking positions. Thus, the pilot would constantly be informed of his position relative to the optimum release conditions. Systems of this nature are presently in operation, however

equipment costs are extremely high. The small cost of the micro-computer would make on board aircraft implementation possible.

APPENDIX A PROGRAM FLOW CHARTS AND PROCESS GRAPHS

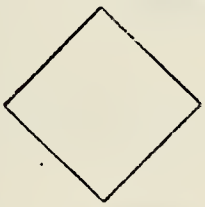
Symbols Used In Flow Charts

Start/Stop

Input/Output

Card Input

Processing Symbol
(Arithmetic Operation)



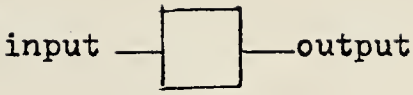
Conditional Symbol

Subroutine Call
(ENTRY) - Entry Point

Return Statement

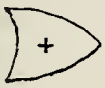
Program Entry Point (label)

Symbols Used in Process Graphs



1. Logical Gates:

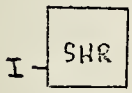
AND Gate



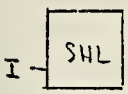
OR Gate (inclusive)

XOR Gate (exclusive)

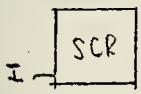
2. Shifting:



Shift Right I Bits



Shift Left I Bits

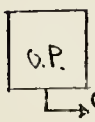


Shift Carry Bit into Word I Bit(s) Right

3. Bite Operations (8 bits):

ADD

MULTIPLY



SUBTRACT

DIVIDE

ASSIGNMENT

4. Address Operations (16 bits):

Full 16 Bit ADD

Full 16 Bit SUBTRACT

Returns 16 Bit Address of Variable

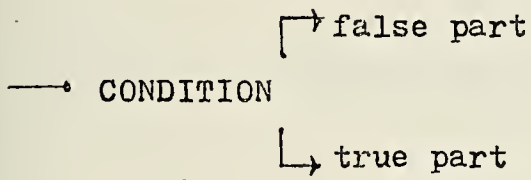
5. Conversion Operations:

Converts Byte to Address Word (upper 8 bits → 0)

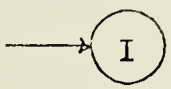
Converts Leftmost 8 Bits of Address Word to Byte Word.

Converts Rightmost 8 Bits of Address Word to Byte Word

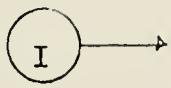
6. Conditonal:



7. Branching:



Destination



Entry Point

8. Floating Point Operations:

ADD (Subtract) Routine

MULTIPLY Routine

DIVIDE Routine

SQUARE ROOT Routine

9. Subroutine Calls

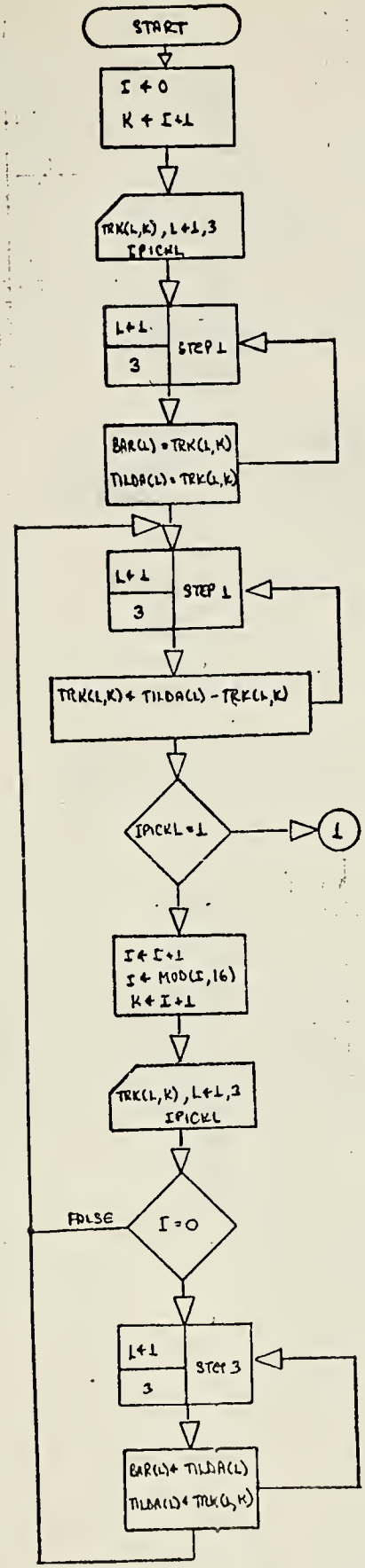
Two Hexidecimal Table Look-Up (MULTIPLY Routine)

Floating Point Table Look-Up Routine

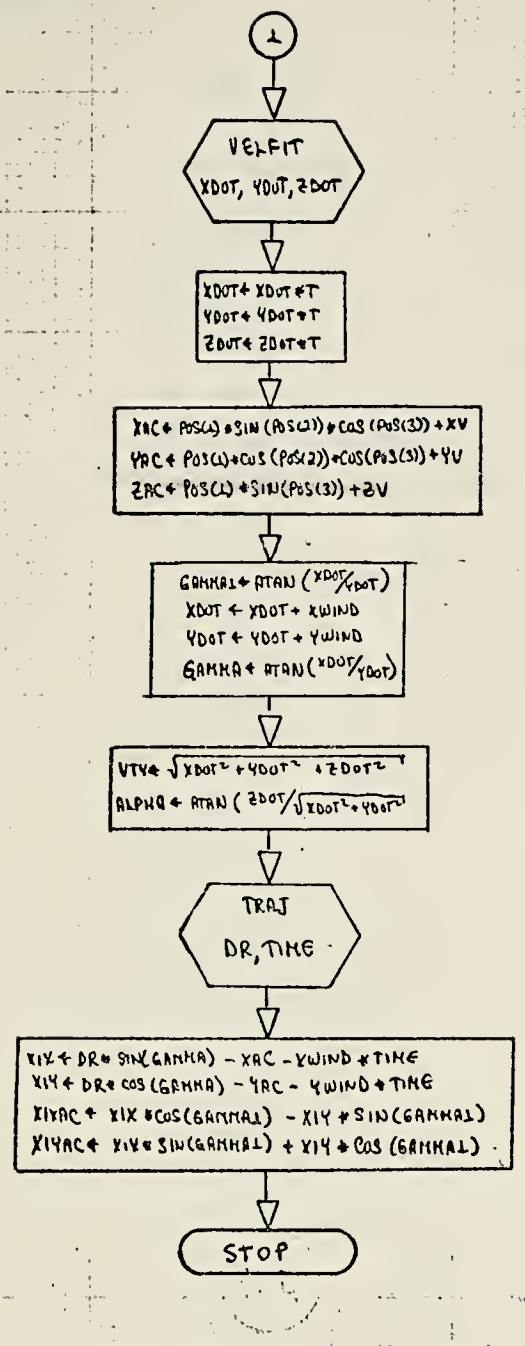
Routine to Calculate First Derivate of Input

Runge-Kutta Integration Routine

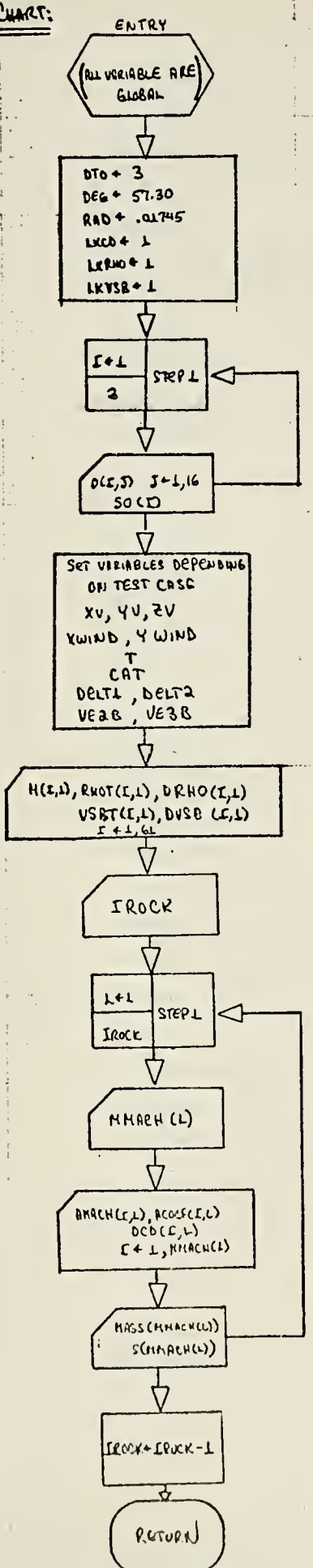
MAIN PROGRAM:



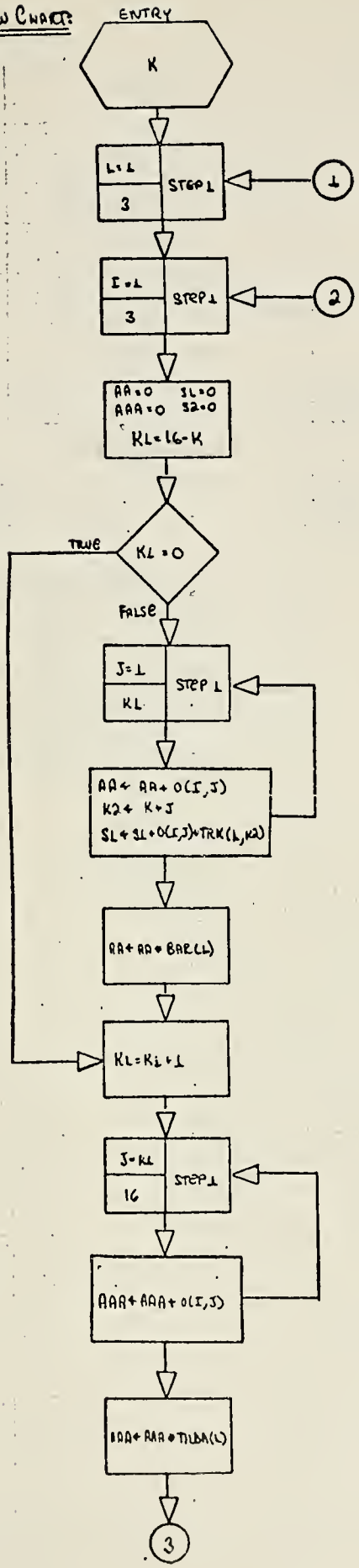
MAIN (CONT.) 1


1
VELFIT
XDOT, YDOT, ZDOT
XDOT ← XDOT * T
YDOT ← YDOT * T
ZDOT ← ZDOT * T
XAC ← POS(1) + SIN(POS(2)) * COS(POS(3)) + XV
YAC ← POS(1) * COS(POS(2)) * COS(POS(3)) + YV
ZAC ← POS(1) * SIN(POS(3)) + ZV
GAMMA1 ← ATAN(XDOT/YDOT)
XDOT ← XDOT + XWIND
YDOT ← YDOT + YWIND
GAMMA ← ATAN(XDOT/YDOT)
VTV ← √(XDOT² + YDOT² + ZDOT²)
ALPHA ← ATAN(ZDOT/√(XDOT² + YDOT²))
TRAJ
DR, TIME
XIX ← DR * SIN(GAMMA) - XAC - XWIND * TIME
XIY ← DR * COS(GAMMA) - YAC - YWIND * TIME
XIYAC ← XIX * COS(GAMMA1) - XIY * SIN(GAMMA1)
XIYAC ← XIX * SIN(GAMMA1) + XIY * COS(GAMMA1)
STOP

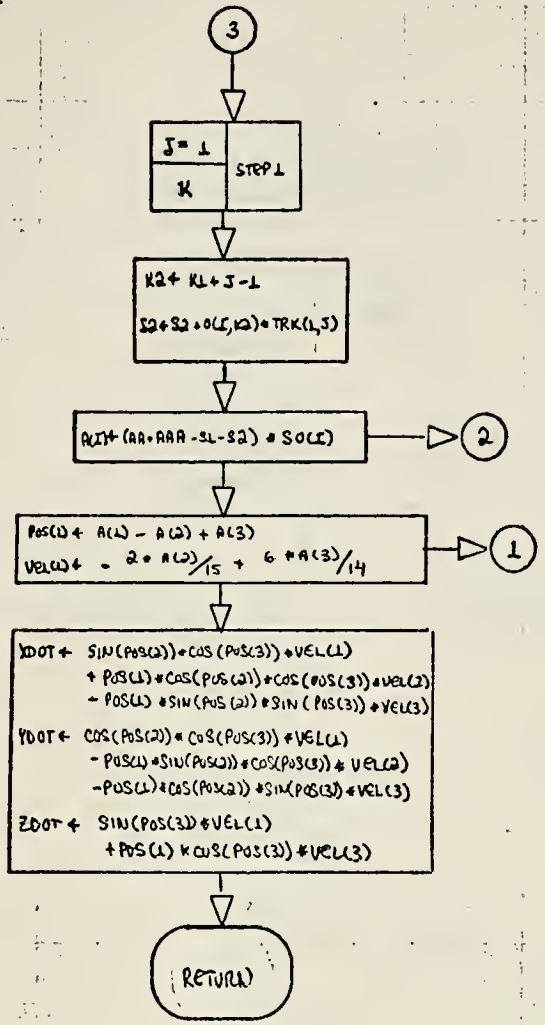
INPUT SUBROUTINE FLOW CHART:



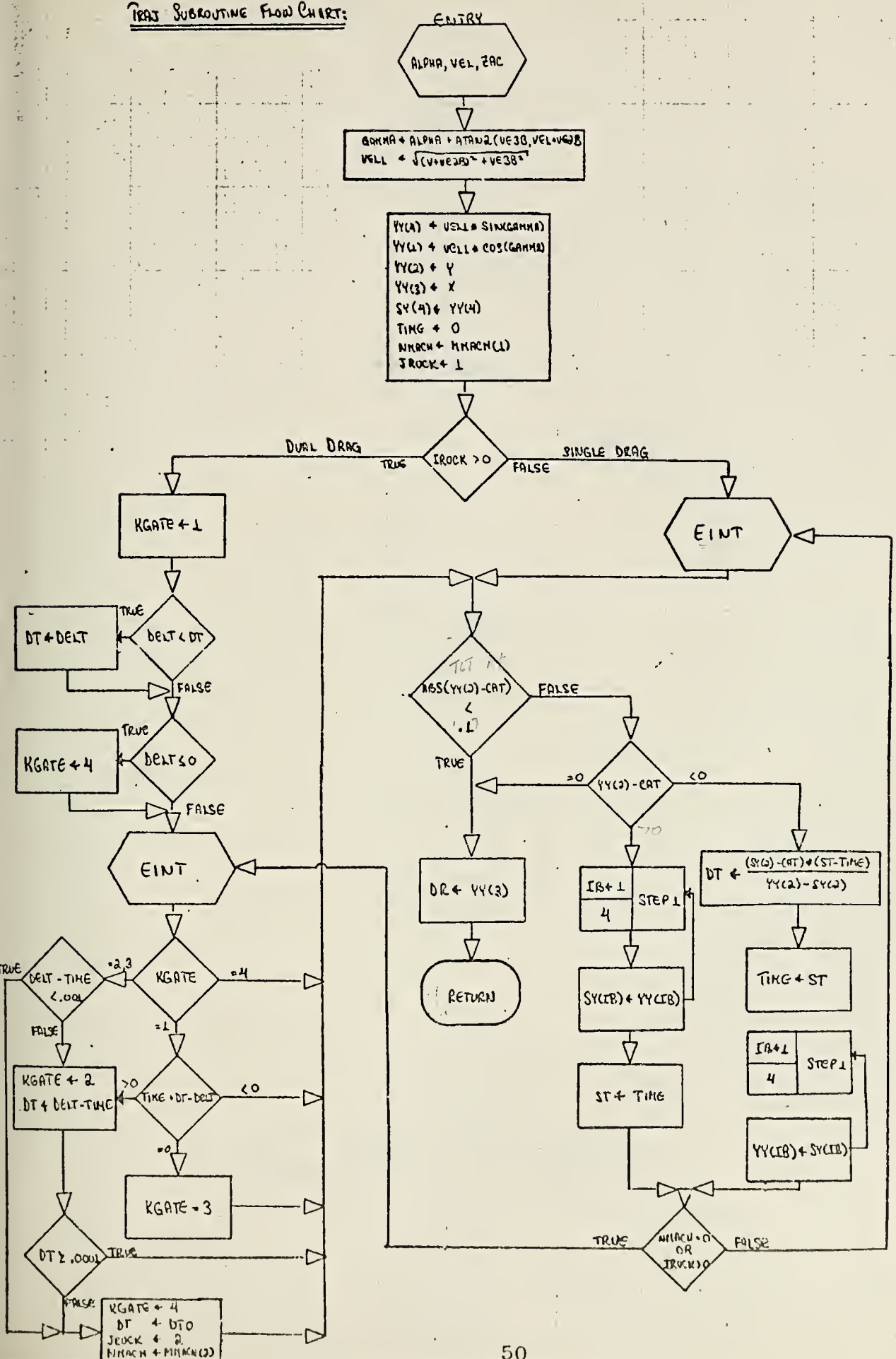
VELFIT Subroutine Flow Chart:



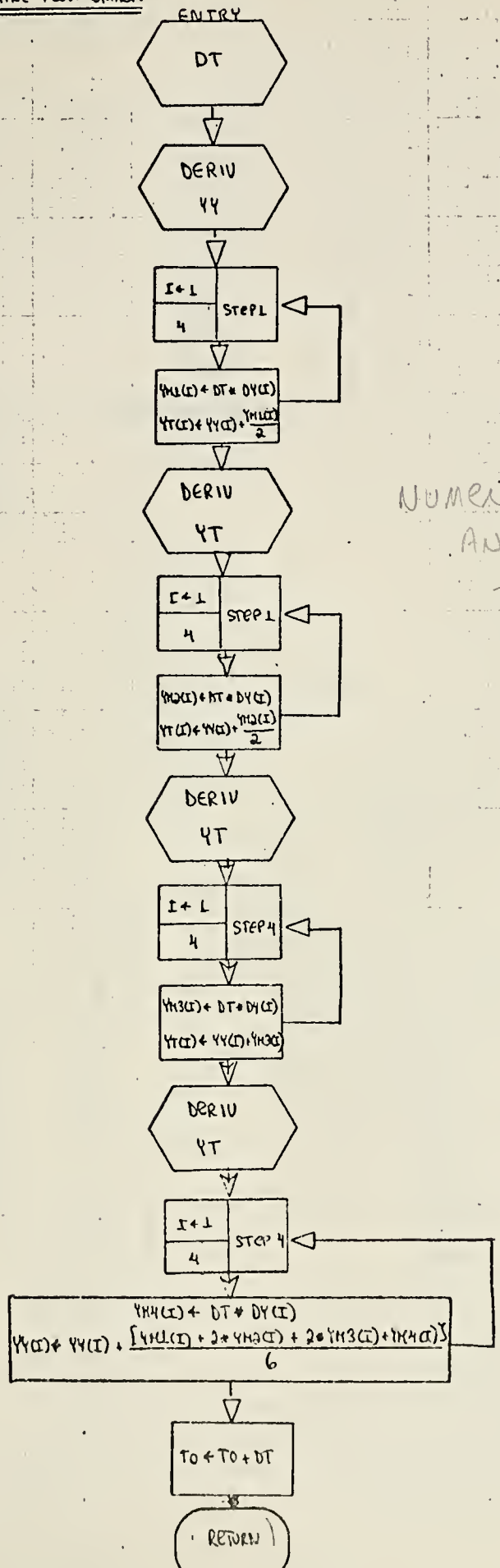
VELFIT (CONT.):


3
J = 1
K
STEP 1
K2 ← K1 + J - 1
S2 ← S2 + O(I,K2) * TRK(I,J)
A(I) ← (AA * AAA - S1 - S2) * SO(I)
2
POS(I) ← A(1) - A(2) + A(3)
VEL(I) ← - 2 * A(2)/15 + 6 * A(3)/14
1
XDOT ← SIN(POS(2)) * COS(POS(3)) * VEL(1)
+ POS(1) * COS(POS(2)) * COS(POS(3)) * VEL(2)
- POS(1) * SIN(POS(2)) * SIN(POS(3)) * VEL(3)
YDOT ← COS(POS(2)) * COS(POS(3)) * VEL(1)
- POS(1) * SIN(POS(2)) * COS(POS(3)) * VEL(2)
- POS(1) * COS(POS(2)) * SIN(POS(3)) * VEL(3)
ZDOT ← SIN(POS(3)) * VEL(1)
+ POS(1) * COS(POS(3)) * VEL(3)
RETURN

TRAJ SUBROUTINE FLOW CHART:

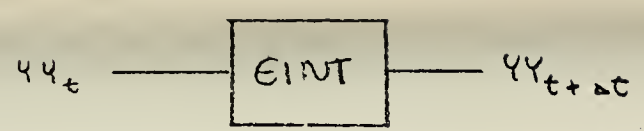


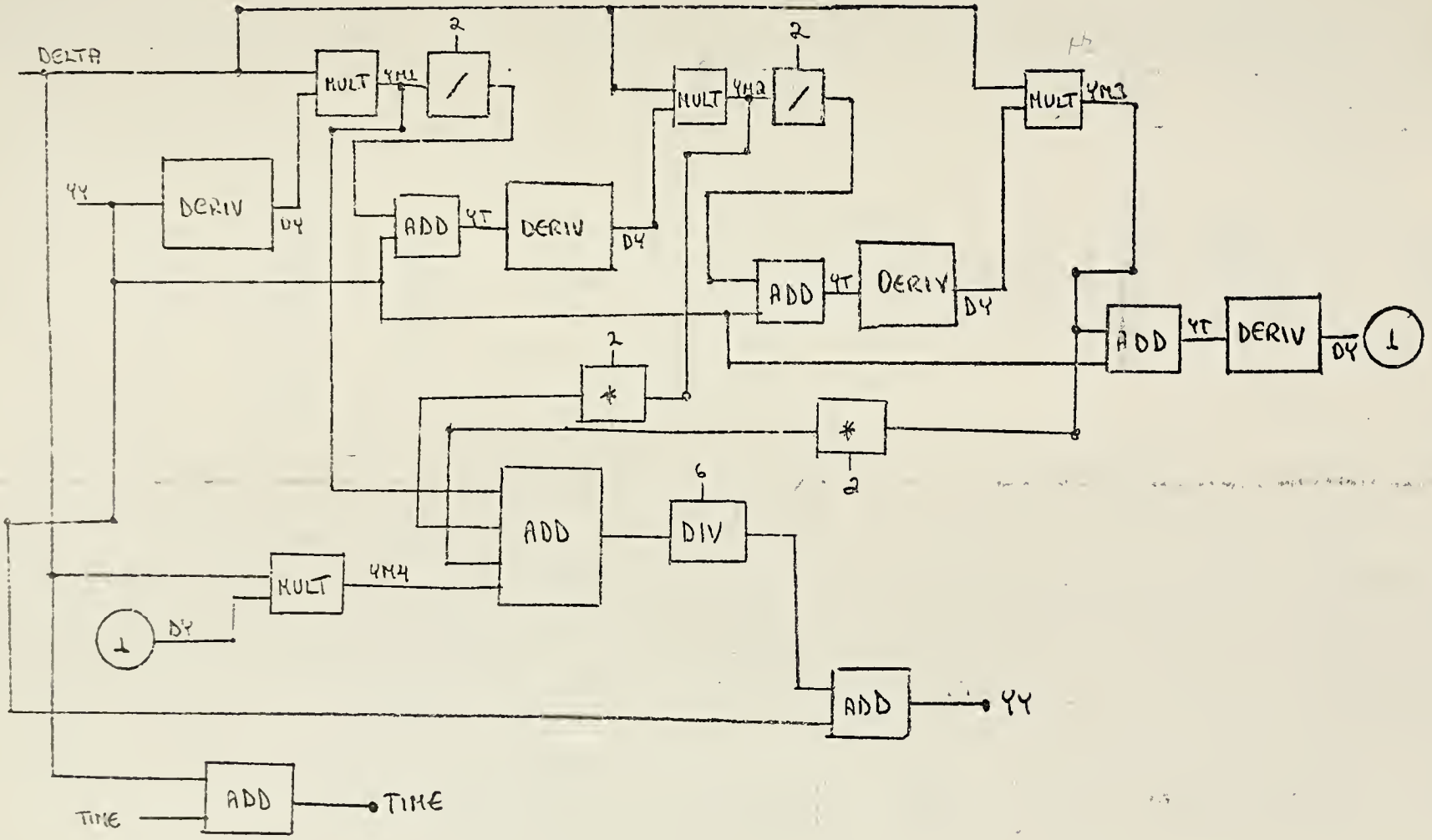
EINT SUBROUTINE FLOW CHART



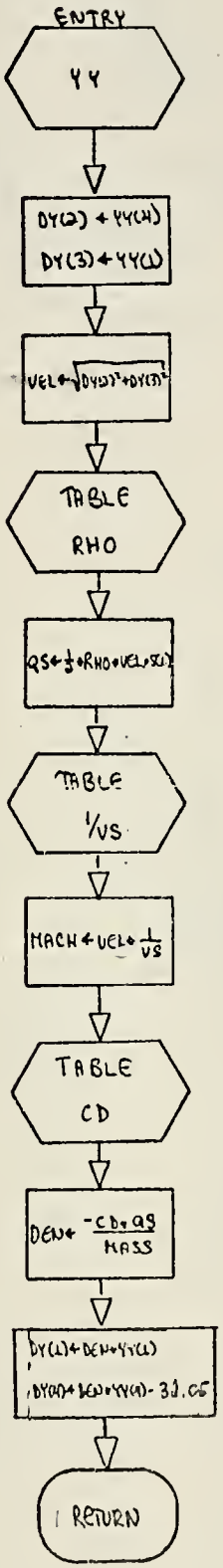
NUMERICAL ANALYSIS 337

EINT PROCESS CHART:





DERIV Subroutine Flow Chart:



DERIV PROCESS CHART

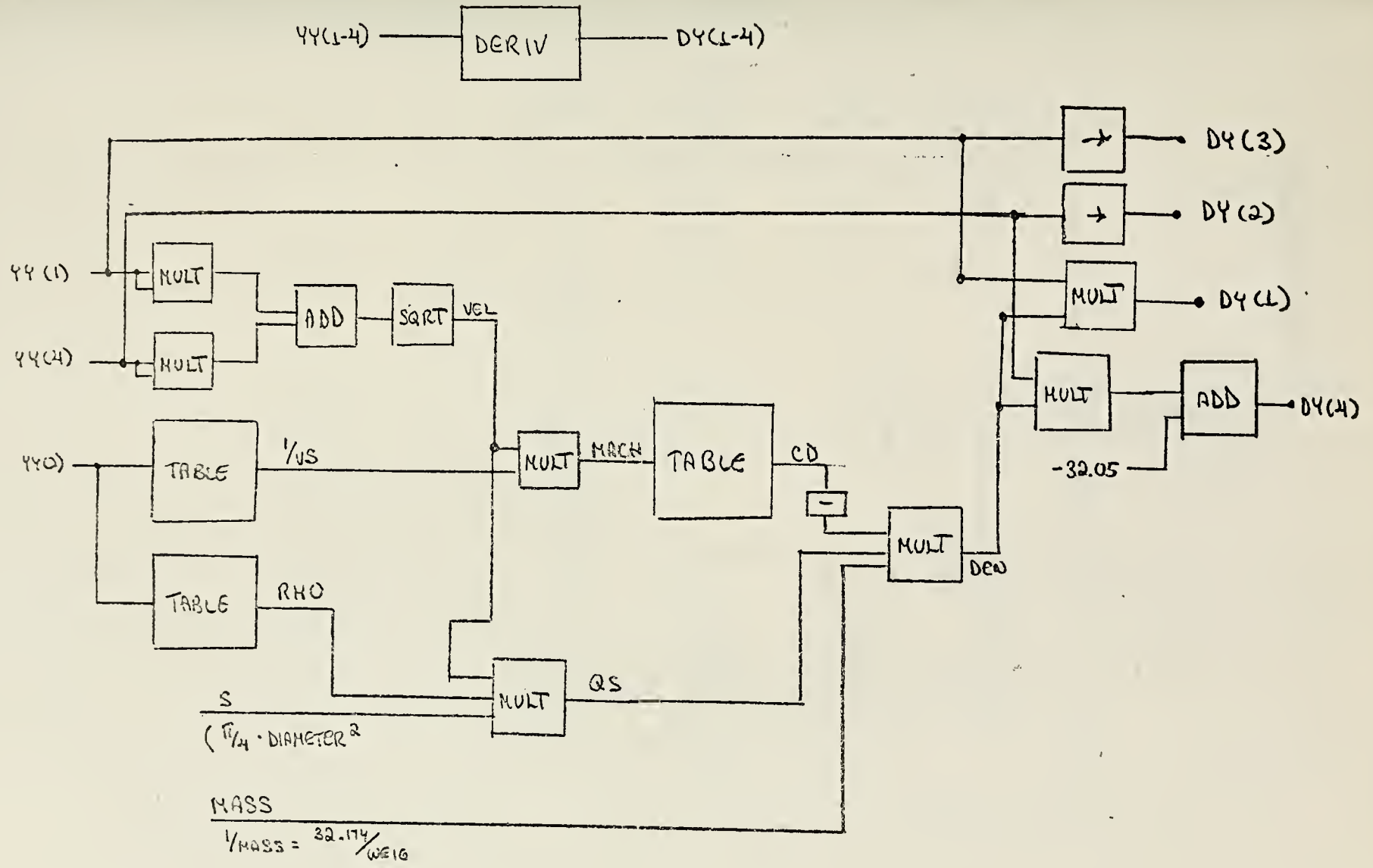
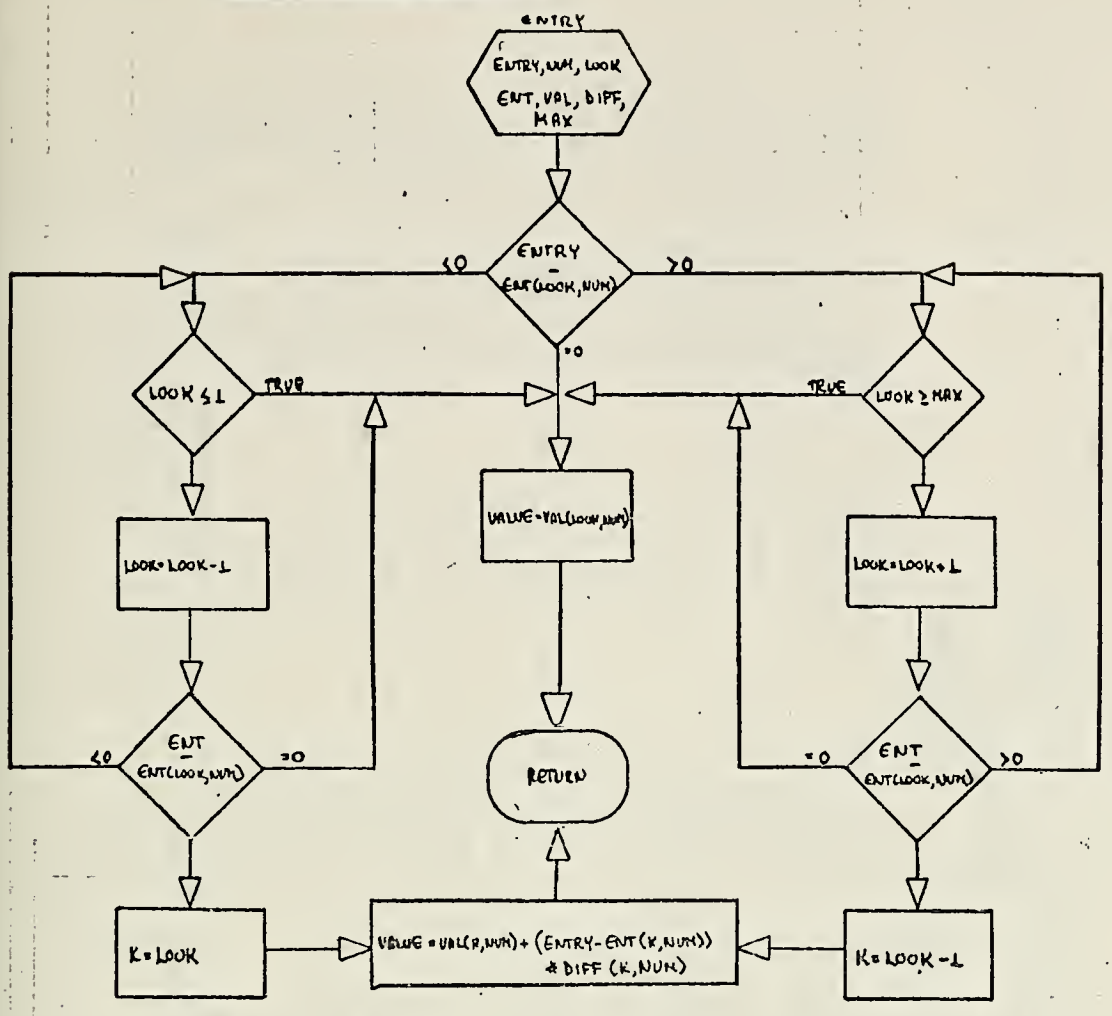


TABLE SUBROUTINE FLOW CHART:



FORTRAN RESULTS

WPN11 MK83 MOD FUZE

FORTRAN TEST CASES (INTEGRATION SCHEME, TIME INTERVAL)

RELEASE CONDITIONS	PUB RESULT Δt, DR	EULER Δt = .01 sec	RUNGA KUTTA Δt = 1 sec	RUNGA KUTTA Δt = 2 sec	RUNGA KUTTA Δt = 3 sec	RUNGA KUTTA Δt = 4 sec	RUNGA KUTTA Δt = 5 sec
#1 α = +10° V = 350 Z = 1000 ft	11.71 sec 6701 ft	11.54 (1154) 6675.35	-1.45% -.38%	11.53 (17) 6673.31	-1.54% -.41%	11.53 (11) 6673.14	-1.54% -.42%	11.53 (9) 6673.13	-1.54% -.42%	11.53 (8) 6673.10	-1.54% -.42%	11.53 (8) 6673.03	-1.54% -.42%
#2 α = +10° V = 350 Z = 9000	17.29 sec 9826 ft	17.11 (1711) 9872.55	-1.04% .47%	17.11 (21) 9873.18	-1.04% .48%	17.11 (14) 9873.26	-1.04% .48%	17.11 (11) 9873.25	-1.04% .48%	17.11 (10) 9873.25	-1.04% .48%	17.11 (9) 9873.21	-1.04% .48%
#3 α = +10° V = 650 Z = 500	13.89 sec 14192 ft	13.82 (1382) 14729.18	-.50% 3.76%	13.82 (19) 14724.34	-.50% 3.75%	13.82 (12) 14723.61	-.50% 3.75%	13.82 (10) 14723.53	-.50% 3.75%	13.82 (9) 14723.37	-.50% 3.75%	13.82 (10) 14723.61	-.50% 3.76%
#4 α = +10° V = 655 Z = 3000	20.85 sec 20859	20.71 (2071) 21923.41	-.67% 5.10%	20.70 (24) 21925.55	-.72% 5.11%	20.70 (16) 21924.91	-.72% 5.11%	20.70 (12) 21925.30	-.72% 5.11%	20.70 (11) 21924.71	-.72% 5.11%	20.69 (10) 21926.50	-.76% 5.13%
#5 α = 0° V = 300 Z = 2600	14.85 sec 7382 ft	14.67 (1467) 7374.15	-1.21% -.11%	14.67 (18) 7374.30	-1.21% -.10%	14.67 (13) 7374.39	-1.21% -.10%	14.67 (10) 7374.39	-1.21% -.10%	14.67 (9) 7374.38	-1.21% -.10%	14.67 (8) 7374.38	-1.21% -.10%
#6 α = 0° V = 300 Z = 15000	30.98 sec 15164 ft	30.63 (3063) 15294.40	-1.13% .86%	30.63 (34) 15305.54	-1.13% .93%	30.63 (21) 15305.59	-1.13% .93%	30.63 (16) 15305.60	-1.13% .93%	30.63 (13) 15305.61	-1.13% .93%	30.63 (12) 15305.62	-1.13% .93%
#7 α = 0° V = 450 Z = 5000	17.82 sec 13127 ft	17.60 (1760) 13198.96	-1.23% .55%	17.59 (22) 13201.36	-1.29% .57%	17.59 (14) 13201.45	-1.29% .57%	17.59 (11) 13201.45	-1.29% .57%	17.59 (10) 13201.45	-1.29% .57%	17.59 (9) 13201.43	-1.29% .57%
#8 α = -20° V = 460 Z = 5500	12.81 sec 7963 ft	12.65 (1265) 7936.68	-1.25% -.33%	12.65 (16) 7937.07	-1.25% -.33%	12.65 (12) 7937.40	-1.25% -.32%	12.65 (10) 7937.41	-1.25% -.32%	12.65 (9) 7937.41	-1.25% -.32%	12.65 (8) 7937.39	-1.25% -.32%
#9 α = -20° V = 550 Z = 7000	13.50 sec 11387 ft	13.24 (1324) 11396.98	-1.93% .09%	13.24 (17) 11398.14	-1.93% .10%	13.24 (12) 11398.60	-1.93% .10%	13.24 (10) 11398.61	-1.93% .10%	13.24 (9) 11398.61	-1.93% .10%	13.24 (8) 11398.59	-1.93% .10%
#10 α = -40° V = 460 Z = 5000	8.18 sec 4699 ft	8.09 (809) 4654.70	-1.10% -.73%	8.09 (12) 4654.38	-1.10% -.74%	8.09 (8) 4654.77	-1.10% -.73%	8.09 (8) 4654.77	-1.10% -.73%	8.09 (6) 4654.75	-1.10% -.73%	8.09 (7) 4654.75	-1.10% -.73%
#11 α = -60° V = 400 Z = 7000	9.61 sec 3194 ft	9.52 (952) 3154.33	-.94% -1.24%	9.52 (13) 3154.53	-.94% -1.24%	9.52 (10) 3155.06	-.94% -1.22%	9.52 (9) 3155.06	-.94% -1.22%	9.52 (8) 3155.06	-.94% -1.22%	9.52 (7) 3155.05	-.94% -1.22%

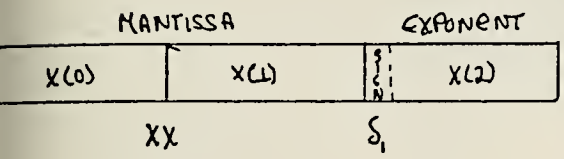
SAMPLE ENTRY

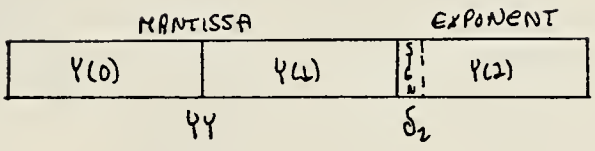
TIME OF FALL (SEC) — $\frac{\text{TEST}\Delta t - \text{PUB}\Delta t}{\text{PUB}\Delta t}$

(# OF CALLS TO THE INTEGRATION ROUTINE)

DOWN RANGE TRAVEL (FT) — $\frac{\text{TEST(DR)} - \text{PUB(DR)}}{\text{PUB(DR)}}$

FLOATING POINT ADD ROUTINE:





GENERAL PROCEDURE:

$X = \pm m_1 \cdot 2^p \rightarrow \delta_1 m_1 \cdot 2^p$

$Y = \pm m_2 \cdot 2^q \rightarrow \delta_2 m_2 \cdot 2^q$

m_1 & m_2 → 16 binary digits

THUS $X+Y = \delta_1 m_1 \cdot 2^p + \delta_2 m_2 \cdot 2^q$

$= \delta_1 m_1 \cdot 2^{p_1+q} + \delta_2 m_2 \cdot 2^q$

$= \delta_1 m_1 \cdot 2^p + \delta_2 m_2 \cdot 2^{p-t}$

$= (\delta_1 m_1 + \delta_2 m_2 2^{-t}) \cdot 2^p$

ASSUME: $p > q$

$p - q = p_1$

$p = p_1 + q$

$q - p = -t \quad t \geq 0$

$q = p - t$

$\therefore \quad \delta_1 (m_{15}^1 \; m_{14}^1 \; \overset{15-c}{\downarrow} \; m_{13}^1 \; \ldots\ldots\ldots \; m_0^1) \rightarrow \delta_1 XX$

$\delta_2 (0 \; 0 \; m_{15}^2 \; \ldots\ldots\ldots \; m_{15-c}^2) \rightarrow \delta_2 YY$

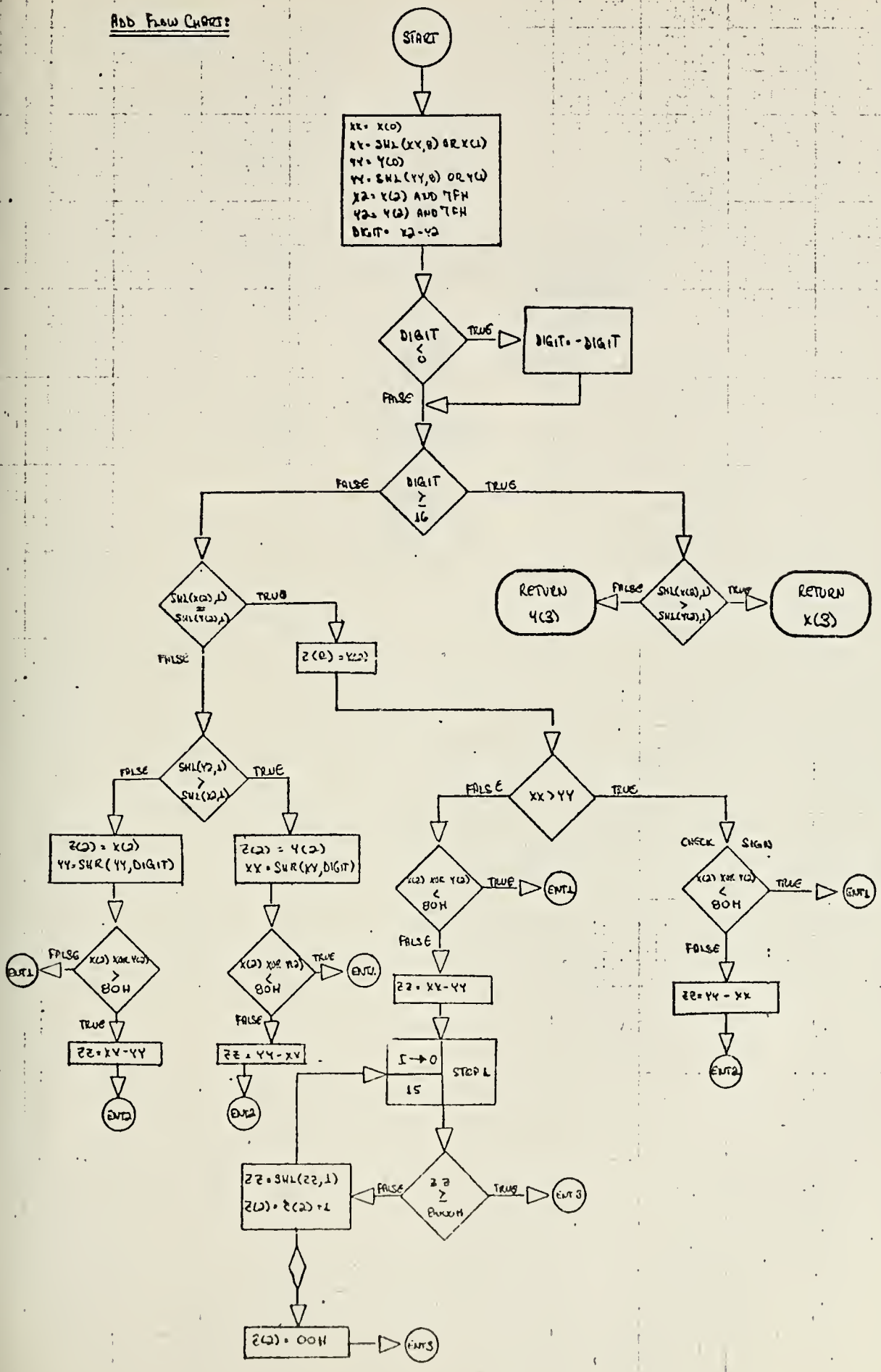
$(z_{15} \; z_{14} \; z_{13} \; \ldots\ldots\ldots \; z_0)$

ZZ

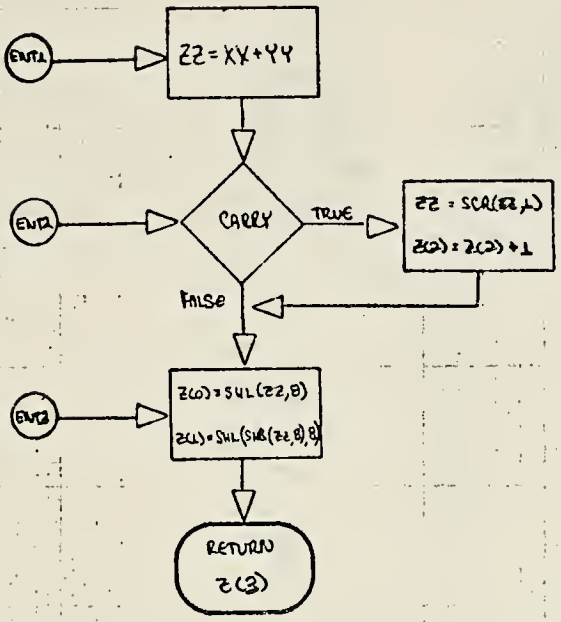
IF $\delta_1 = \delta_2$ THEN ADD AND ATTACH δ_1 TO SUM

$\delta_1 \neq \delta_2$ THEN SUBTRACT AND ATTACH δ_1 TO DIFFERENCE

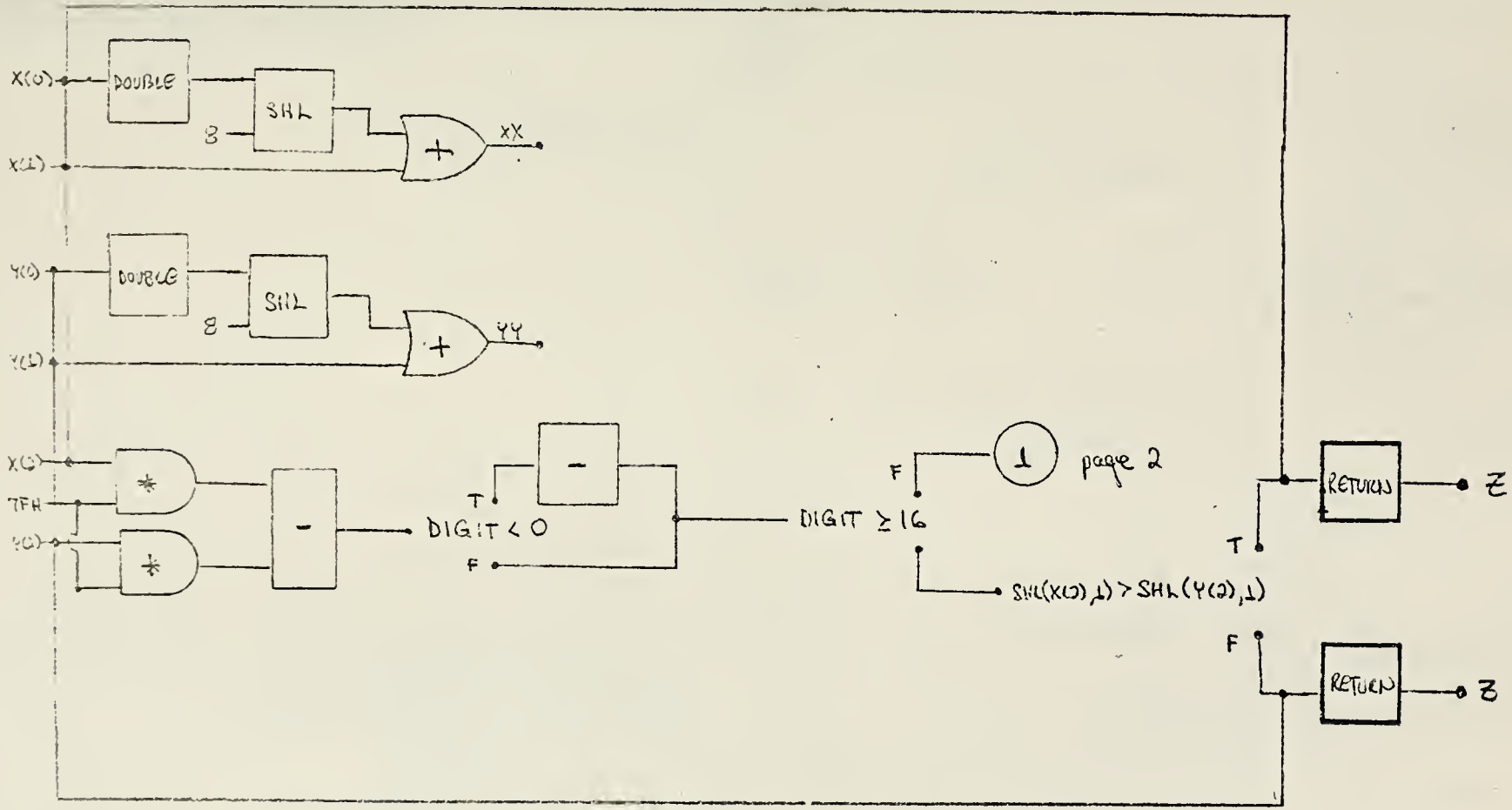
ADD FLOW CHART:

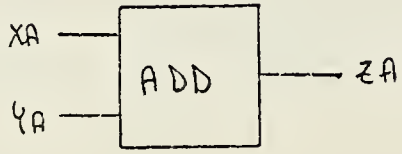


ADD (CONT.)

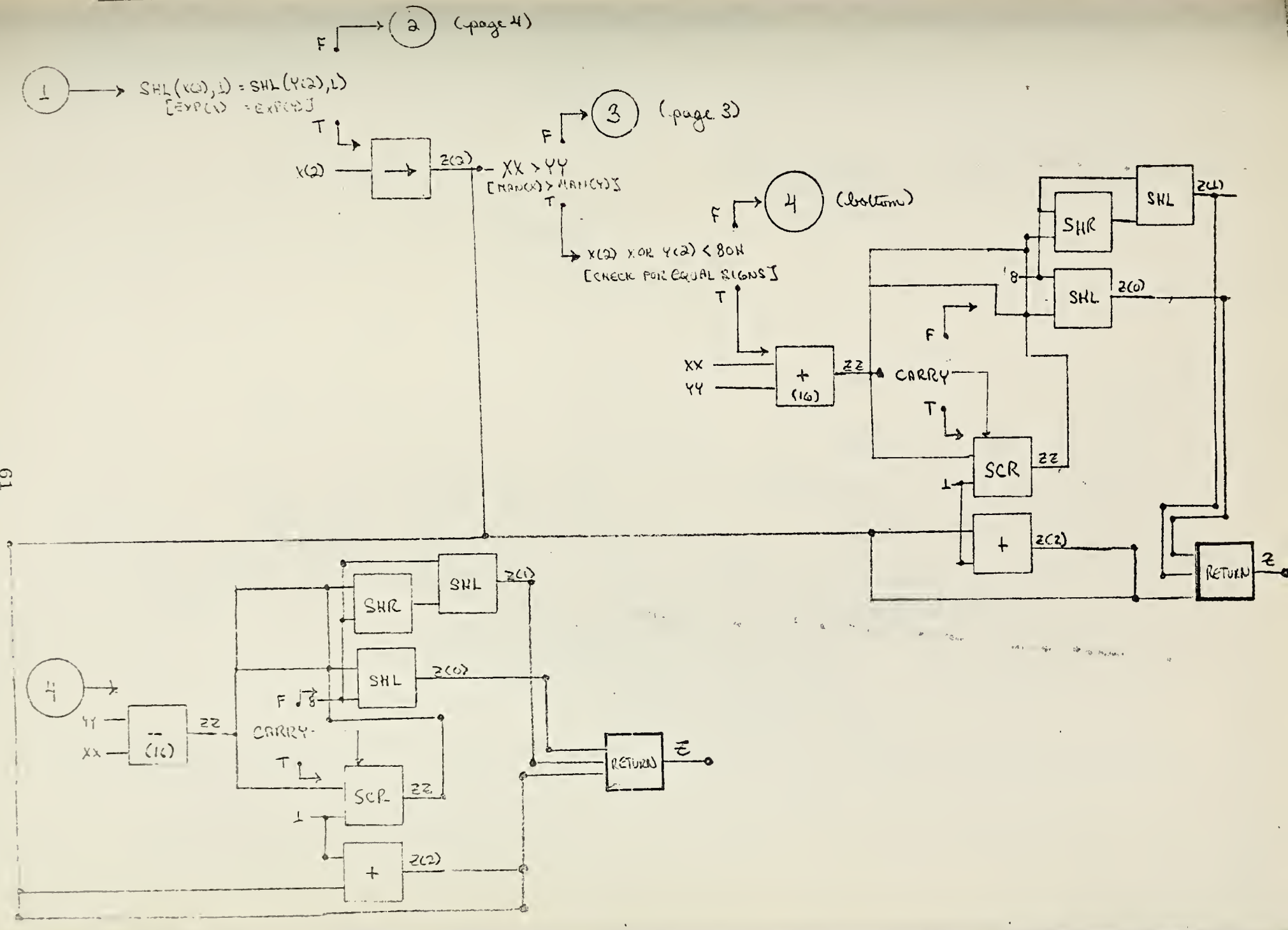

ENTRY
ZZ = XX + YY
ENTRY
CARRY
TRUE
ZZ = SCR(ZZ,1)
Z(2) = Z(2) + 1
FALSE
ENTRY
Z(0) = SHL(ZZ,8)
Z(1) = SHL(SHR(ZZ,8),8)
RETURN
Z(3)

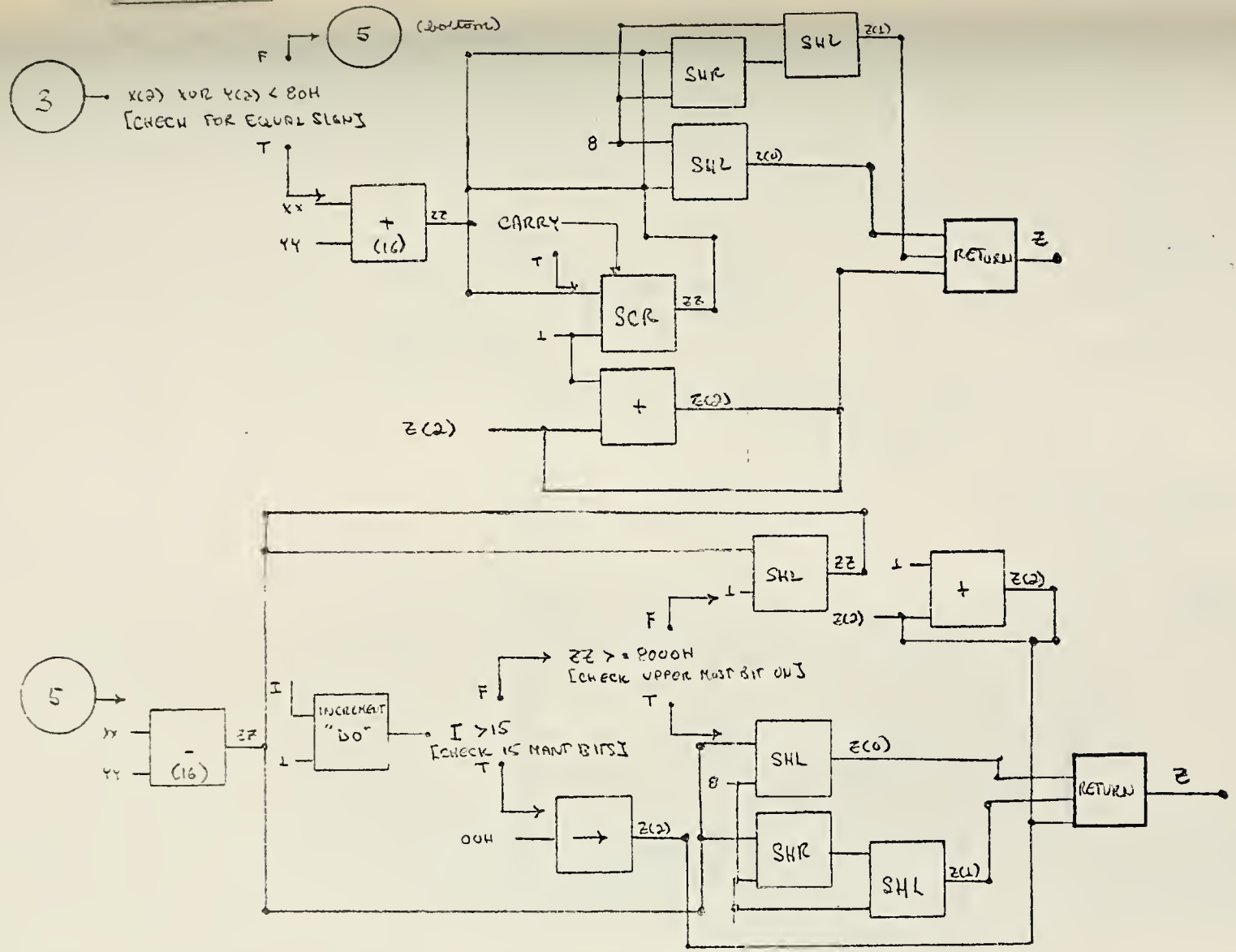
ADD PROCESS CHART:

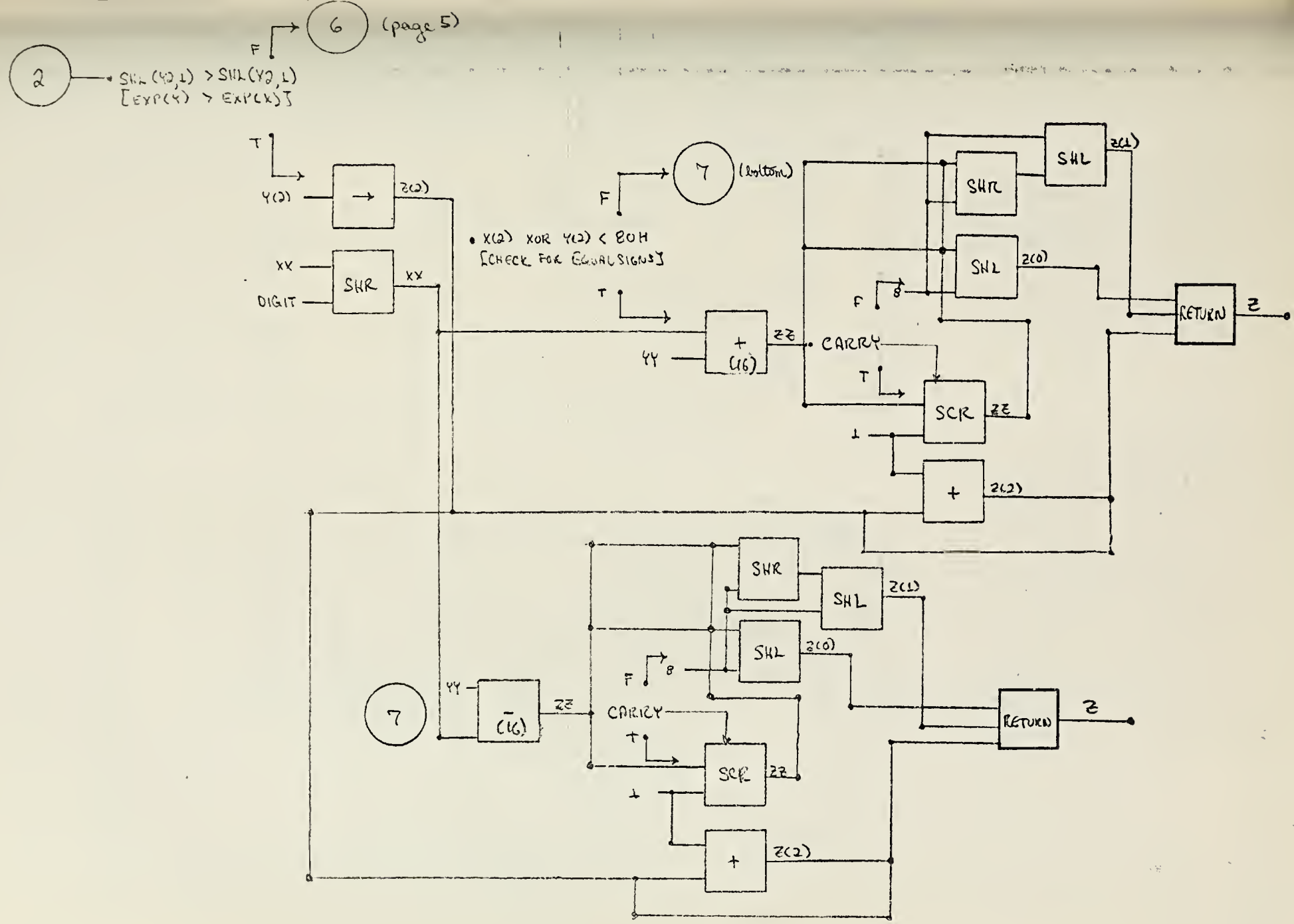


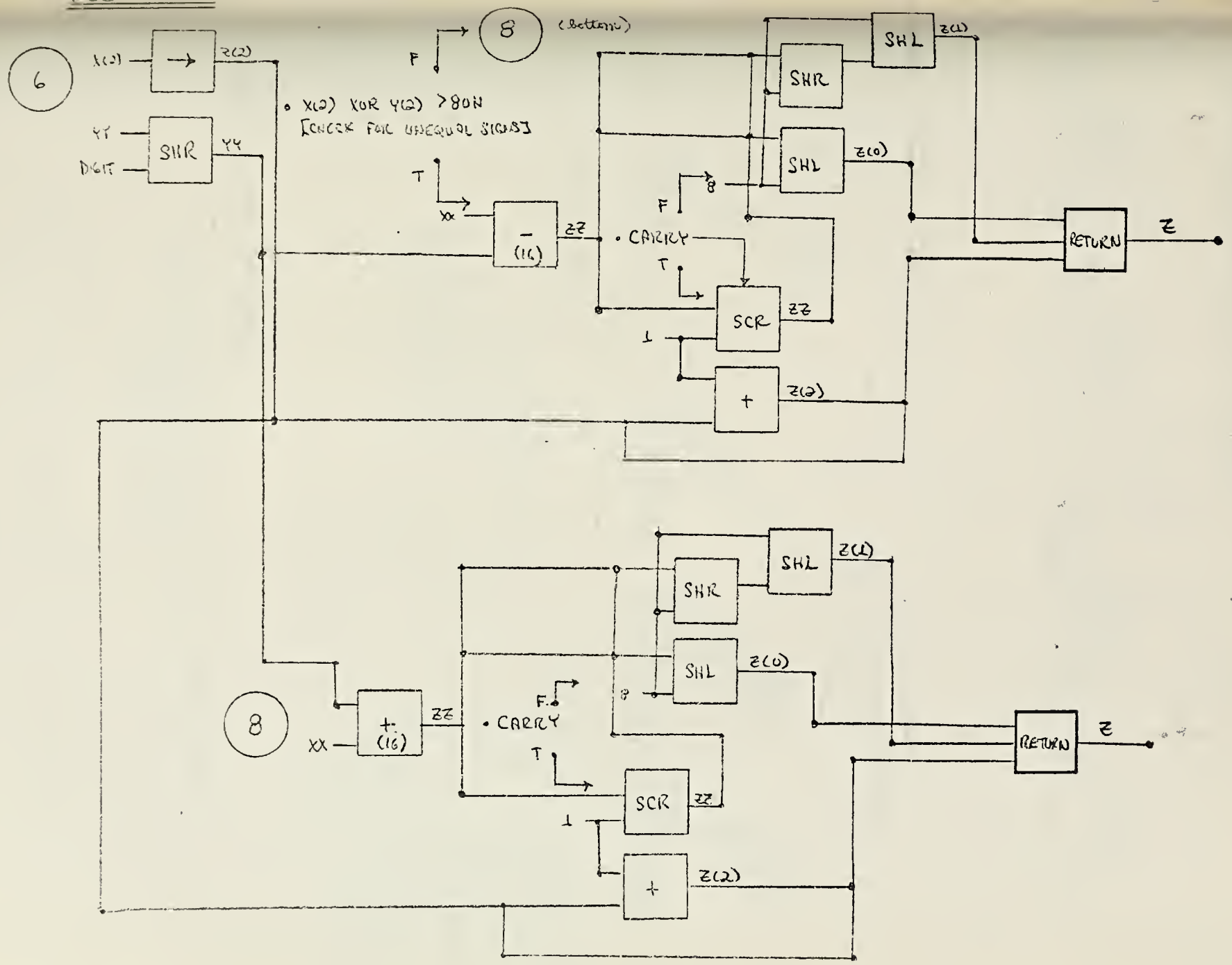


ADD (CONT.):

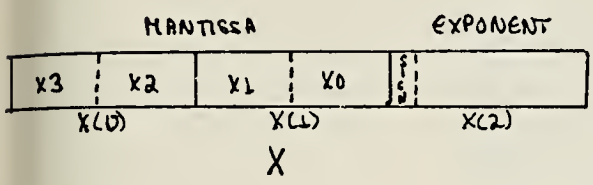

F → (2) (page 4)
(1) → SHL(X(0),1) = SHL(Y(2),L)
[EXP(X) = EXP(Y)]
F → (3) (page 3)
T
X(2)
Z(3)
XX > YY
[MAN(X) > MAN(Y)]
T
X(2) XOR Y(2) < 80H
[CHECK FOR EQUAL SIGNS]
F → (4) (bottom)
T
XX
YY
+ (16)
ZZ
CARRY
F
8
SHR
SHL
Z(1)
SHL
Z(0)
T
SCR
ZZ
⊥
+
Z(2)
RETURN
Z
(4)
YY
XX
-- (16)
ZZ
CARRY
F
SHR
SHL
Z(1)
SHL
Z(0)
T
SCR
ZZ
⊥
+
Z(2)
RETURN
Z

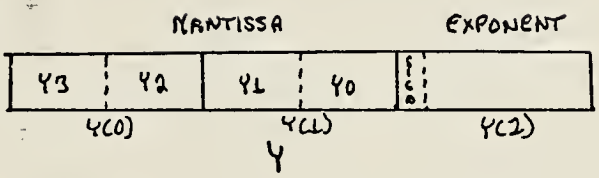

3
X(2) XOR Y(2) < 80H
[CHECK FOR EQUAL SIGN]
F
5
(bottom)
T
XX
YY
+
(16)
ZZ
CARRY
SHR
SHL
Z(1)
8
Z(0)
RETURN
Z
SCR
Z(2)
5
INCREMENT
"DO"
I
I >15
[CHECK IS MANT BITS]
ZZ >= 8000H
[CHECK UPPER MOST BIT ON]
-
(16)
000H


2
SHL(Y2,L) > SHL(Y2,L)
[EXP(Y) > EXP(X)]
F
6
(page 5)
T
Y(2)
→
Z(2)
XX
DIGIT
SHR
XX
F
7
(bottom)
• X(2) XOR Y(2) < 80H
[CHECK FOR EQUAL SIGNS]
T
YY
+
(16)
ZZ
CARRY
SHR
SHL
Z(1)
SHL
Z(0)
F
8
RETURN
Z
T
SCR
ZZ
1
+
Z(2)
SHR
SHL
Z(1)
SHL
Z(0)
F
8
7
YY
−
(16)
ZZ
CARRY
RETURN
Z
T
SCR
ZZ
1
+
Z(2)


6
X(2)
Z(2)
YY
DIGIT
SHR
YY
• X(2) XOR Y(2) >80N
[CHECK FOR UNEQUAL SIGNS]
T
XX
−
(16)
ZZ
F
8
8 (bottom)
• CARRY
T
SCR
ZZ
1
+
Z(2)
SHR
SHL
Z(1)
SHL
Z(0)
RETURN
Z
8
XX
+
(16)
ZZ
F.
• CARRY
T
SCR
ZZ
1
+
Z(2)
SHR
SHL
Z(1)
SHL
Z(0)
RETURN
Z

FLOATING POINT MULTIPLY ROUTINE:





STEP 1: MULTIPY MANTISSA

$$\begin{array}{r} Y_3\ Y_2\ Y_1\ Y_0 \\ *\ X_3\ X_2\ X_1\ X_0 \\ \hline \overline{Y_0 X_0} \\ \overline{Y_1 X_0} \\ \overline{Y_2 X_0} \\ \overline{Y_3 X_0} \\ \hline Z_4^0\ Z_3^0\ Z_2^0\ Z_1^0\ Z_0^0 \end{array}$$

NOTE $\overline{Y_j X_i}$ DENOTES TWO HEX DIGIT PRODUCT

$$\begin{array}{r} \overline{Y_0 X_1} \\ \overline{Y_1 X_1} \\ \overline{Y_2 X_1} \\ \overline{Y_3 X_1} \\ \hline Z_4^1\ Z_3^1\ Z_2^1\ Z_1^1\ Z_0^1 \end{array} \qquad \begin{array}{r} \overline{Y_0 X_2} \\ \overline{Y_1 X_2} \\ \overline{Y_2 X_2} \\ \overline{Y_3 X_2} \\ \hline Z_4^2\ Z_3^2\ Z_2^2\ Z_1^2\ Z_0^2 \end{array} \qquad \begin{array}{r} \overline{Y_0 X_3} \\ \overline{Y_1 X_3} \\ \overline{Y_2 X_3} \\ \overline{Y_3 X_3} \\ \hline Z_4^3\ Z_3^3\ Z_2^3\ Z_1^3\ Z_0^3 \end{array}$$

SUM OF PARTIALS:

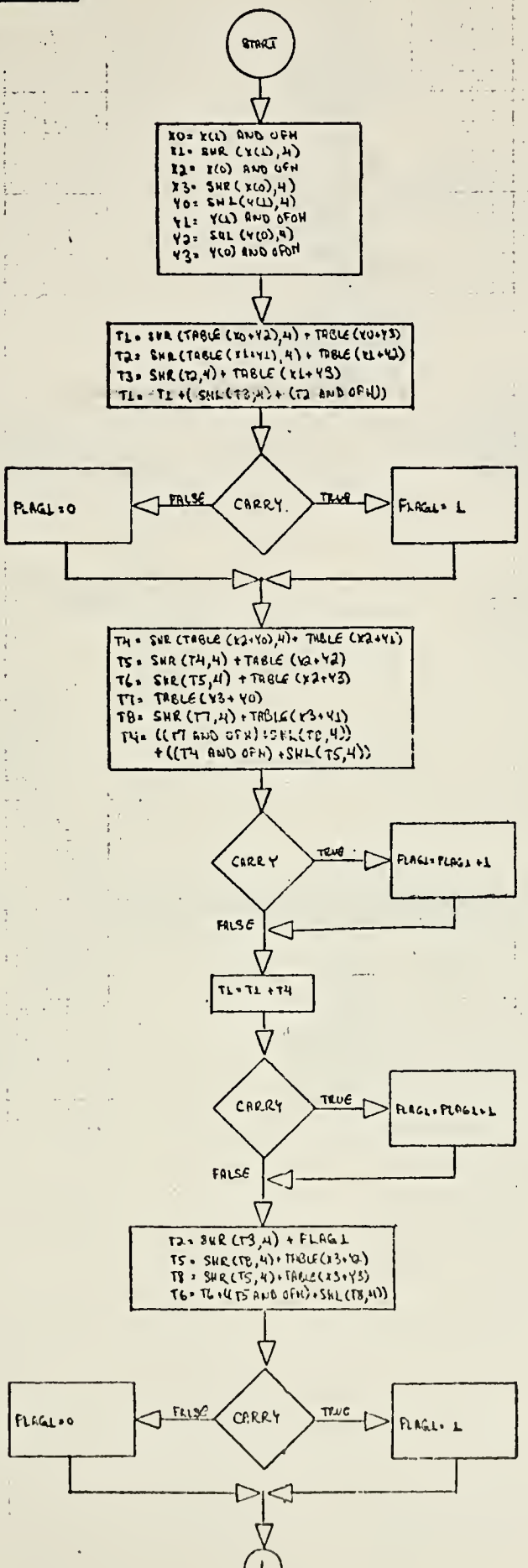
$$\begin{array}{rrrrr|rrr} & & Z_4^0 & Z_3^0 & & Z_2^0 & Z_1^0 & Z_0^0 \\ & Z_4^1 & Z_3^1 & Z_2^1 & & Z_1^1 & Z_0^1 & \\ Z_4^2 & Z_3^2 & Z_2^2 & Z_1^2 & & Z_0^2 & & \\ Z_4^3 & Z_3^3 & Z_2^3 & Z_1^3 & Z_0^3 & & & \\ \hline Z_7 & Z_6 & Z_5 & Z_4 & Z_3 & Z_2 & Z_1 & Z_0 \end{array}$$

NOTE: EACH Z_i^j CONTAINS 4 BITS (1 HEX DIGIT OF INFO) THUS WE ARE CONCERNED WITH ONLY 5 HEX DIGITS. SINCE THE MANTISSA CONTAINS 4 HEX DIGIT, THE 5th IS USED FOR ROUNDING.

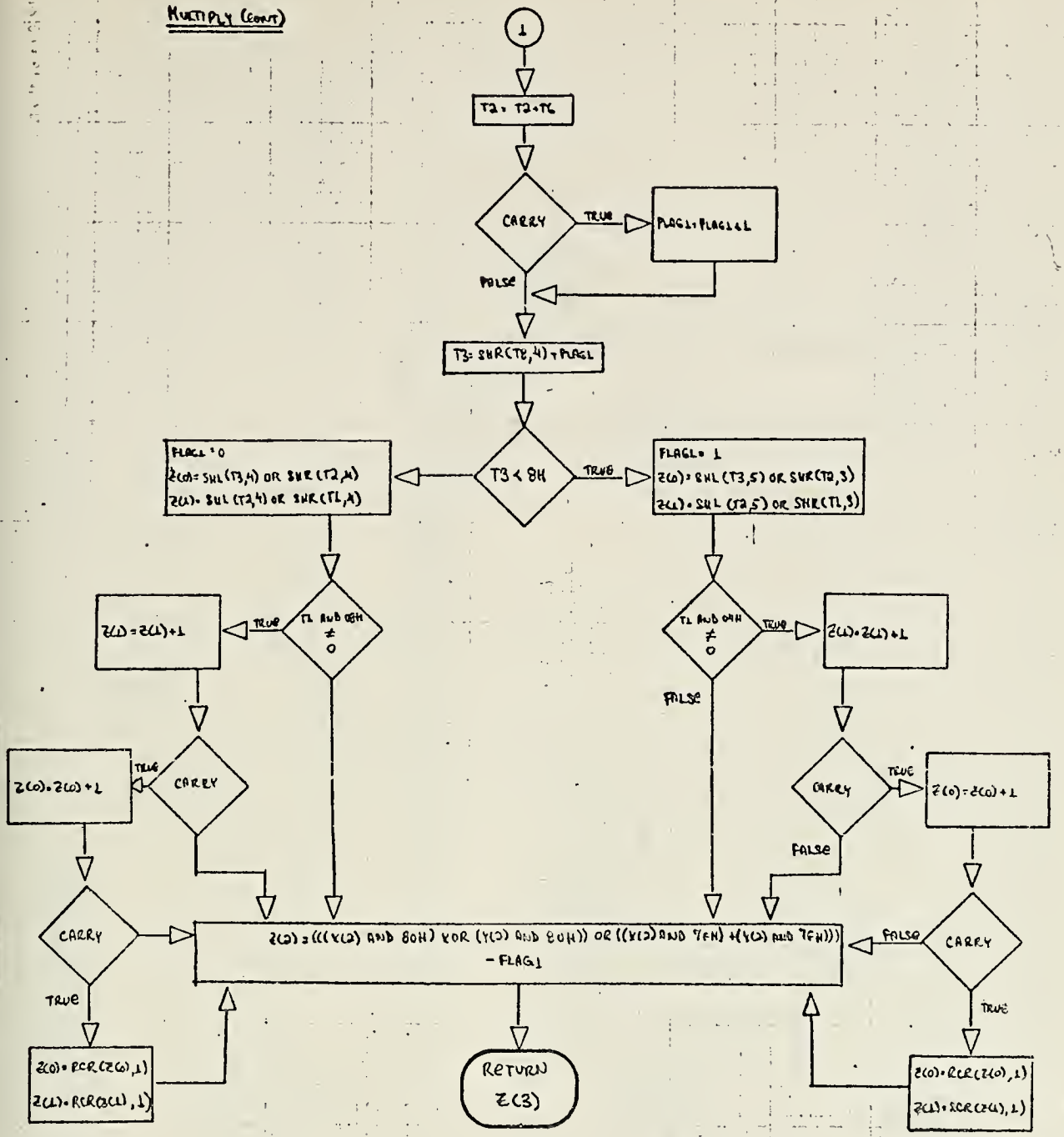
STEP 2: ADD EXPONENTS

ADD X(2) + Y(2) AND CHECK SIGN BIT (MANTISSA) FOR PRODUCT SIGN.

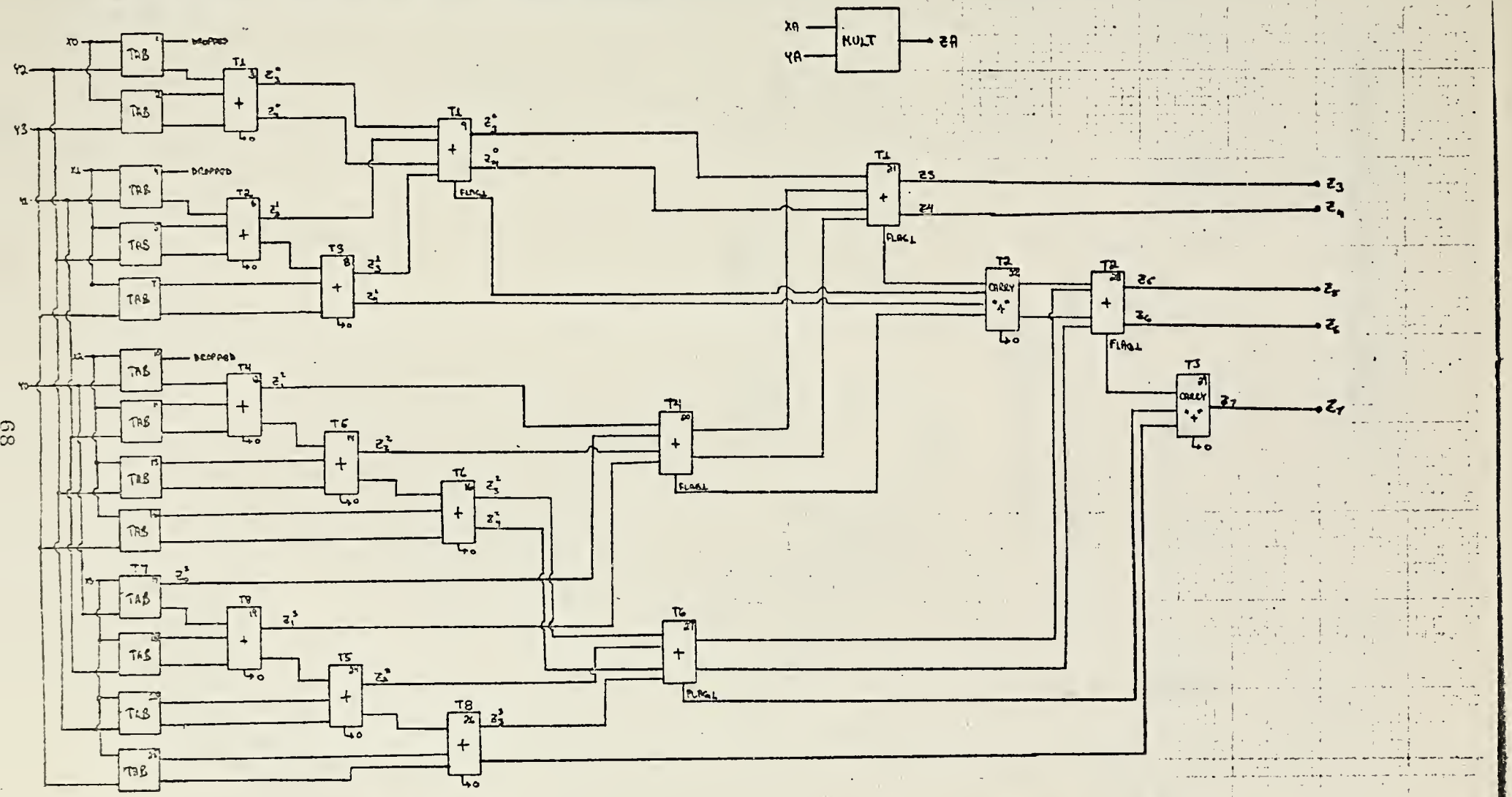
MULTIPLY FLOW CHART:



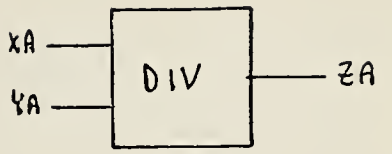
MULTIPLY (CONT)

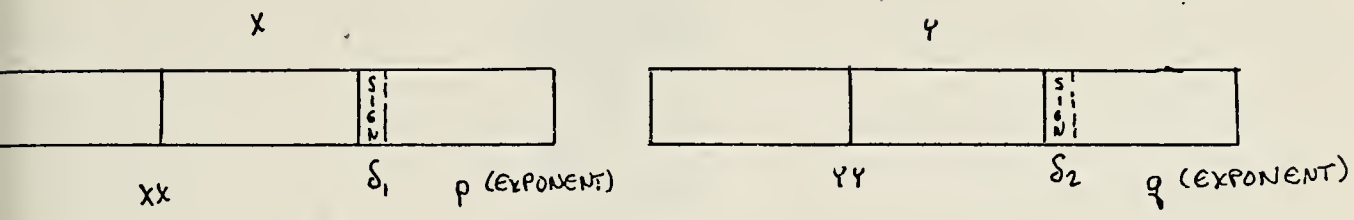


Process Chart For Floating Point Multiply Procedure:



FLOATING POINT DIVIDE ROUTINE:





$$\frac{X}{Y} = \frac{\delta_1 \cdot XX \cdot 2^p}{\delta_2 \cdot YY \cdot 2^q}$$

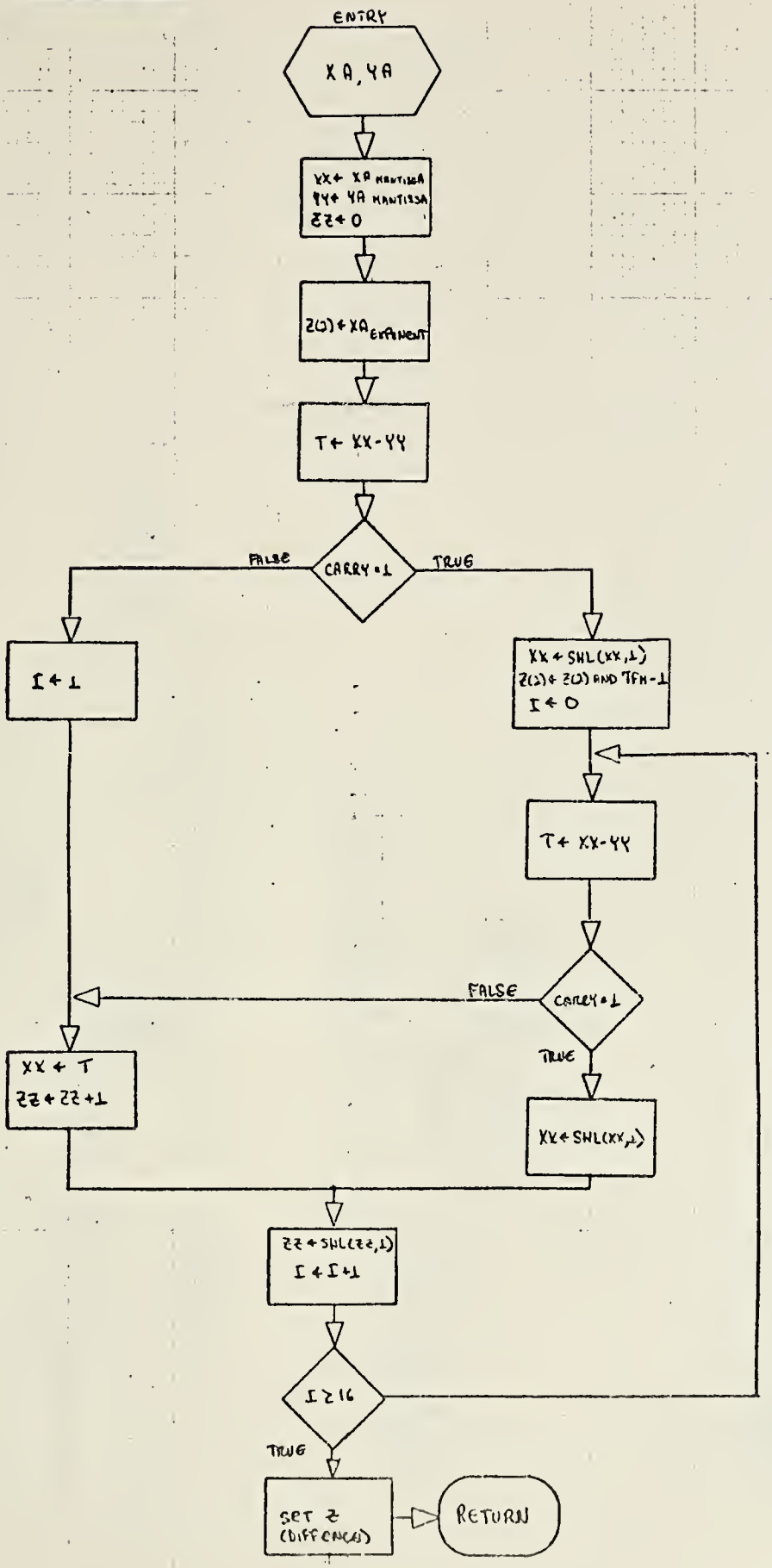
SUBTRACT XX - YY

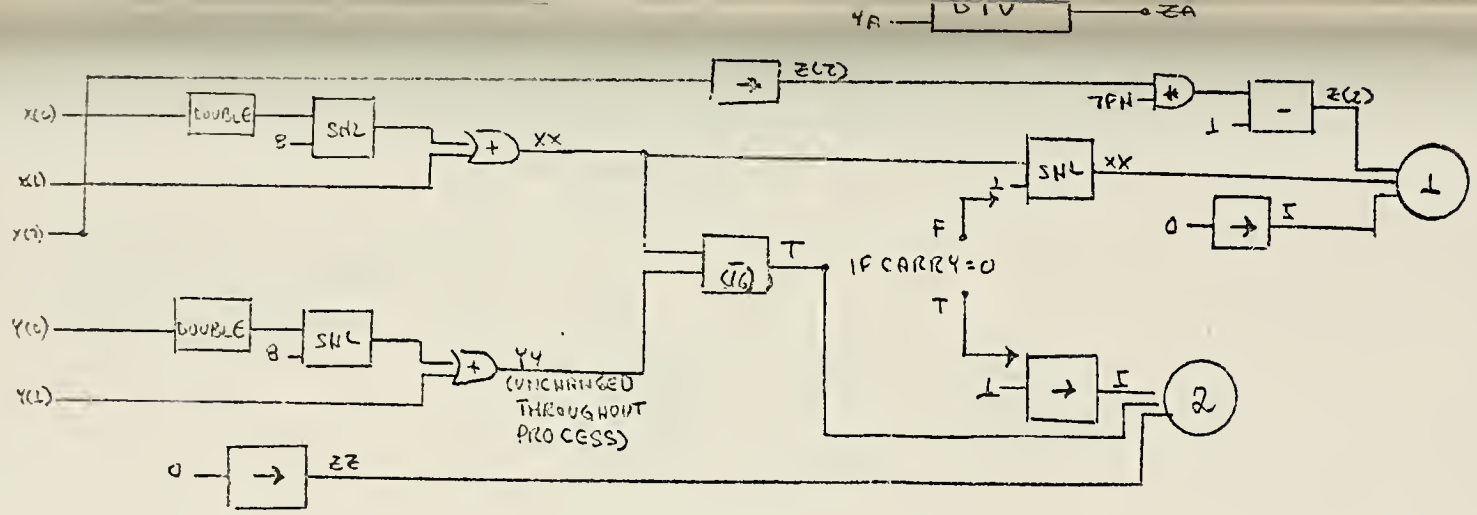
$$= \delta_1 \text{ XOR } \delta_2 \quad p_0\, p_1\, p_2 \cdots p_{15}\; 2^{p-q} \text{ if } p_0 = 1$$

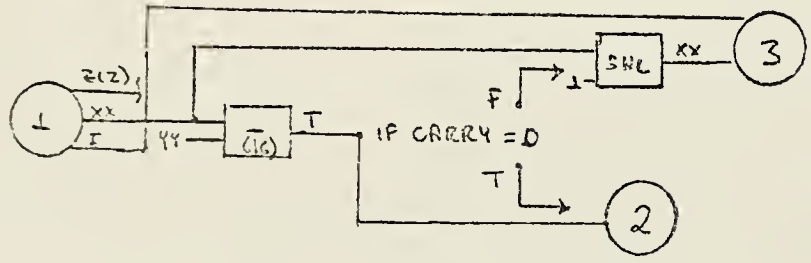
$$\delta_1 \text{ XOR } \delta_2 \quad p_1\, p_2\, p_3 \cdots p_{16}\; 2^{p-q-1} \text{ if } p_0 = 0$$

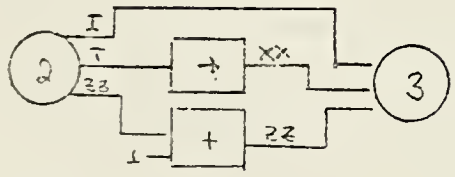
PROCEDURE CONTINUED SHIFTING XX LEFT 1 BIT THROUGH 16 BITS.

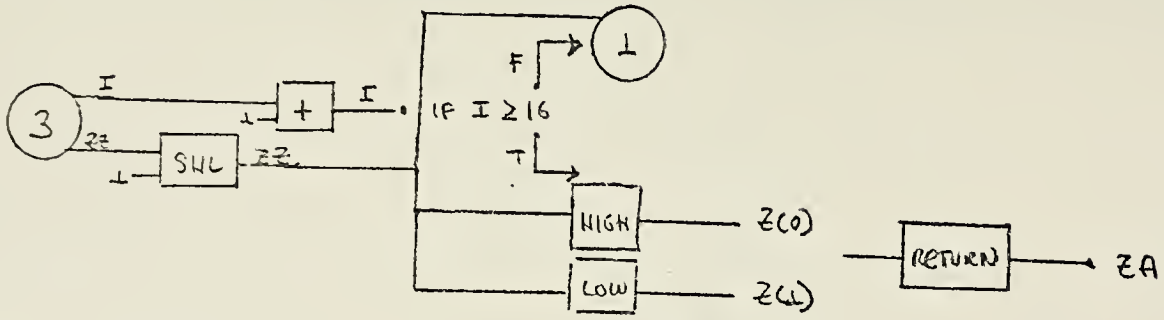
DIVIDE FLOW CHART:




YA
DIV
ZA
X(0)
X(1)
X(2)
DOUBLE
8
SHL
+
XX
Z(2)
7FH
*
1
-
Z(2)
SHL
XX
F
1
0
I
1
(16)
T
IF CARRY=0
T
1
I
2
Y(0)
Y(1)
DOUBLE
8
SHL
+
YY
(UNCHANGED
THROUGHOUT
PROCESS)
0
ZZ


1
Z(2)
XX
I
YY
(16)
T
IF CARRY=0
T
2
SHL
XX
3
F
1


2
I
T
ZZ
1
+
XX
ZZ
3


3
I
1
+
I
IF I ≥ 16
F
1
T
SHL
ZZ
HIGH
Z(0)
LOW
Z(1)
RETURN
ZA

FLOATING POINT SQUARE ROOT ROUTINE:

CALLING ARGUEMENT

SQRT(X)

$$\therefore f(x) = x^2 - C = 0$$

$$f'(x) = 2x$$

USING NEWTON'S ITERATION METHOD

$$X_{n+1} = X_n - \frac{f(x_n)}{f'(x_n)}$$

$$= X_n - \frac{X_n^2 - C}{2X_n}$$

$$= \frac{X_n}{2} + \frac{C}{2X_n}$$

$$= \frac{1}{2}\left(X_n + \frac{C}{X_n}\right)$$

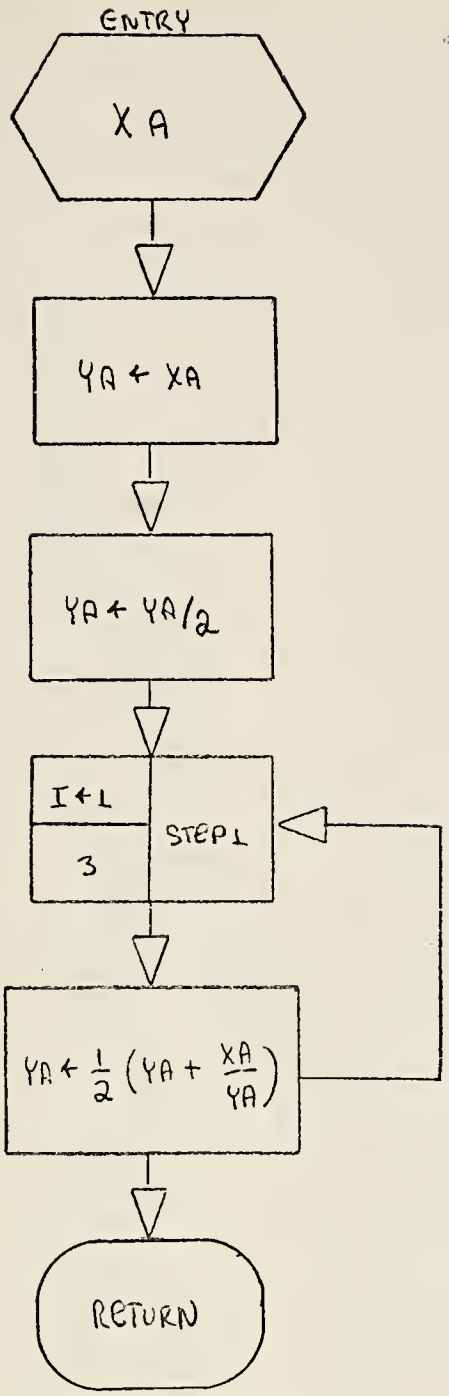
FOR PROGRAM $Y \leftarrow X$

$$Y_{i+1} = \frac{1}{2}\left(Y_i + \frac{X}{Y_i}\right) \qquad i = 0, 1, 2$$

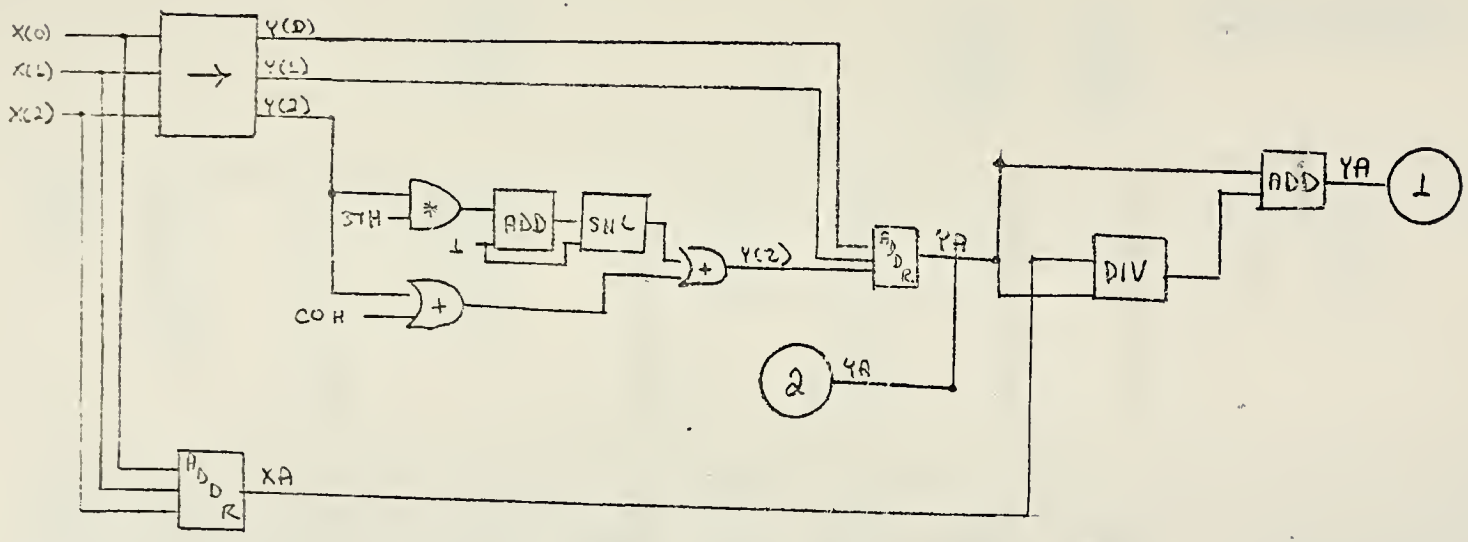
WHERE $Y = SQRT(X)$

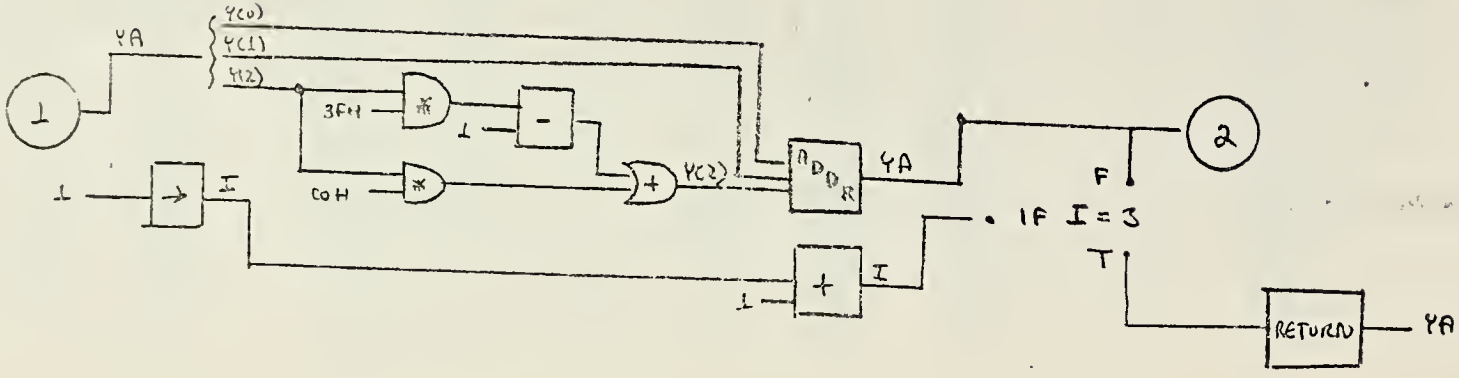
SQUARE ROOT

FLOW CHART :




XA
SQRT
YA


X(0)
X(1)
X(2)
Y(0)
Y(1)
Y(2)
3FH
ADD
SUB
COH
ADDR
YA
DIV
ADD
YA
1
2
YA
ADDR
XA


YA
1
Y(0)
Y(1)
Y(2)
3FH
COH
Y(2)
ADDR
YA
2
F
IF I=3
T
RETURN
YA
1
I
1
I

APPENDIX B PROGRAM DEFINITION OF VARIABLES

SYMBOL FORTRAN	SYMBOL PL/M if varied	UNITS	DEFINITION
A(I)	RA0 RA1 RA2 TA0 TA1 TA2 PA0 PA1 PA2	FT RADIANS RADIANS 	Coefficients calculated using Legendre Polynomials for a Least Squares Position and Velocity Fit. I=1 - A0 I=2 - A1 I=3 - A2 for each r,θ,ϕ
AA	F1SUM1 F1SUM2 F1SUM3	DIMENSIONLESS	Summation of least significant evaluated polynomials in Least Squares Fit.
AAA	F2SUM1 F2SUM2 F2SUM3	DIMENSIONLESS	Summation of most significant evaluated polynomials in Least Squares Fit.
ACOEF		DIMENSIONLESS	Drag coefficient values in inputed Mach vs. Drag table(s).
ALPHA		RADIANS	Pilot's bomb release dive angle.
AMACH		DIMENSIONLESS	Mach number values in inputed Mach vs. Drag table(s).
BAR(L)	BAR1 BAR2 BAR3	FT RADIANS RADIANS	Base values for least recent entries in the position history registers.
CAT		FT	Target altitude
DCD		DIMENSIONLESS	Drag coefficient divided difference values in inputed Mach vs. Drag table(s).
DEG		DEGS/RADS	Radians to degrees conversion constant
DELT		SEC	Total fin opening time for dual drag bombs.

SYMBOL FORTRAN	SYMBOL PL/M if varied	UNITS	DEFINITION
DELT1		SEC	Fin opening time perpendicular to flight path.
DELT2		SEC	Fin opening time parallel to flight path.
DELTA		SEC	Time interval for Runge-Kutta integration.
DEN		1/SEC	Values used in calculating the first derivatives.
DIFF		VARIABLE	Divided difference table values entered in Table look-up routine.
DR		FT	Down range travel of the bomb at impact.
DRHO		SLUGS/FT^3	Density divided difference values in inputed Altitude vs. Density table.
DT		SEC	Time interval for bomb simulation (subject to change during execution).
DTO		SEC	Time interval for bomb simulation (fixed).
DVSB		SEC/FT	Speed of sound (inverse) divided difference values in inputed Altitude vs. 1/Speed of Sound table.
DY(I)	DY1 DY2 DY3 DY4	FT/SEC^2 FT/SEC FT/SEC$_2$ FT/SEC^2	First derivatives of bomb's simulated position and velocity components.
ENT		VARIABLE	Independent table values entered in Table look-up routine.
ENTRY		VARIABLE	Independent variable for Table look-up routine.

SYMBOL FORTRAN	SYMBOL PL/M if varied	UNITS	DEFINITION
GAMMA		RADIANS	Main: aircraft's run-in-line with respect to target's axis. Traj: release angle of the bomb adjusted for ejection velocities.
GAMMA1		RADIANS	Aircraft's wind corrected run-in-line with respect to target's axis.
H		FT	Altitude values inputed for Altitude vs. Density table Altitude vs. 1/Speed of Sound table.
I		DIMENSIONLESS	Pointer used in position history registers.
IPICKL	PICKL	DIMENSIONLESS	Simulated bomb release signal from the aircraft.
IROCK		DIMENSIONLESS	=0 single drag bomb =1 dual drag bomb
JROCK		DIMENSIONLESS	Applicable to dual drag bombs. =1 first stage =2 second stage
K		DIMENSIONLESS	Pointer to release positions in the position history registers.
K1		DIMENSIONLESS	Pointer for the Legendre Polynomials for Least Squares position and velocity fit.
K2		DIMENSIONLESS	Pointer for the position history registers for Least Squares position and velocity fit.

SYMBOL FORTRAN	SYMBOL PL/M if varied	UNITS	DEFINITION
KGATE		DIMENSIONLESS	Applicable to dual drag bombs =1 1st stage (bomb @ release) =2 1st stage to fin opening =3 fin opening time =4 2nd stage
LOOK		DIMENSIONLESS	Inputed pointer to the table location of previously found result in Table look-up routine.
LKCD		DIMENSIONLESS	Pointer to table location of the previously found drag coefficient in the Table look-up routine.
LKRHO		DIMENSIONLESS	Pointer to table location of the previously found density in the Table look-up routine.
LKVSB		DIMENSIONLESS	Pointer to table location of the previously found speed of sound (inverse) value in Table look-up routine.
MACH		DIMENSIONLESS	Mach number of falling bomb (speed in relation to the velocity of sound at a given altitude).
MASS		$1/LB_m$	Inverse of the mass of the bomb.
MAX		DIMENSIONLESS	Inputed number of entries in associated table to be evaluated in Table look-up routine.
MMACH		DIMENSIONLESS	Number of entries in Mach vs. Drag table(s).
NMACH		DIMENSIONLESS	Number of entries in the appropriate Mach vs. Drag table.

SYMBOL FORTRAN	SYMBOL PL/M if varied	UNITS	DEFINITION
NUM		DIMENSIONLESS	Inputed pointer to appropriate Mach vs. Drag table in Table look-up routine.
O(I,16)	01(16) 02(16) 03(16)	DIMENSIONLESS	Evaluated Legendre Polynomials for A0 I=1 A1 I=2 A2 I=3
POS(L)	POS1 POS2 POS3	FT RADIANS RADIANS	Polar release conditions resulting from Least Squares position fit.
QS		SLUGS/SEC	Temporary storage value used in calculating the first derivatives.
RAD		RADS/DEGS	Degrees to Radians conversion constant.
RHOT		SLUGS/FT^3	Density values in inputed Altitude vs. Density table.
S		FT^2	Cross-sectional area of the bomb.
S1	V1SUM1 V1SUM2 V1SUM3	FT RADIANS RADIANS	Summation of the least significant evaluated polynomials multiplied by the corresponding values in the position history register in the Least Squares Fit.
S2	V2SUM1 V2SUM2 V2SUM3	FT RADIANS RADIANS	Summation of the most significant evaluated polynomials multiplied by the corresponding values in the position history register in the Least Squares Fit.

SYMBOL FORTRAN	PL/M if varied	UNITS	DEFINITION
SO(I)	SO1 SO2 SO3	DIMENSIONLESS	Sum of the squares of the evaluated Legendre polynomials for A0 I=1 A1 I=2 A2 I=3
SY(I)	SY1 SY2 SY3 SY4	FT/SEC FT FT FT/SEC	Temporary storage of bomb's simulated positions and velocities. I=1 horizontal velocity I=2 vertical position I=3 horizontal position I=4 vertical velocity
T		1/SEC	Inverse of the tracking time interval.
TILDA(L)	TILDA1 TILDA2 TILDA3	FT RADIANS RADIANS	Base values for most recent entries in the position history registers.
TIME		SEC	Time for bomb to reach impact.
TRK(L,16)	TRK1 TRK2 TRK3	FT RADIANS RADIANS	Position history registers containing the 16 most recent r,θ,ϕ radar output values. Actual values contained are the differences between the actual radar output and the most recent base value.
VAL		VARIABLE	Dependent table values entered in the Table look-up routine.
VALUE		VARIABLE	Dependent variable (output) for the Table look-up routine.
VE2B		FT/SEC	Ejection velocity parallel to the flight path.
VE3B		FT/SEC	Ejection velocity perpendicular to the flight path.

SYMBOL FORTRAN	PL/M if varied	UNITS	DEFINITION
VEL		FT/SEC	Total velocity of the bomb during the trajectory simulation.
VEL(I)	VEL1 VEL2 VEL3	FT/SEC RADIANS/SEC RADIANS/SEC	Polar release velocities resulting from the Least Squares velocity fit.
VSBT		SEC/FT	Speed of Sound (inverse) values in inputed Altitude vs. 1/Speed of Sound table.
VTY		FT/SEC	Total velocity of the aircraft at bomb release.
XAC		FT	Aircraft's X position component at release, with respect to target center.
XDOT		FT/SEC	Aircraft's X velocity component at release.
XIX	IMPX	FT	X impact position with respect to target's run-in-line.
XIXAC	IMPXAC	FT	X impact position with respect to aircraft's run-in-line.
XIY	IMPY	FT	Y impact position with respect to target's run-in-line.
XIYAC	IMPYAC	FT	Y impact position with respect to aircraft's run-in-line.
XV		FT	X distance from van's axis to target center.
XWIND		FT/SEC	X wind component.
XX		FT/SEC	Temporary storage used in calculating aircraft's release dive angle.

SYMBOL FORTRAN	SYMBOL PL/M if varied	UNITS	DEFINITION
YAC		FT	Aircraft's Y position component at release, from target center.
YDOT		FT/SEC	Aircraft's Y velocity component at release.
YM1(I)	YM11	FT/SEC	First estimate in Runge-Kutta Integration.
	YM12	FT	
	YM13	FT	
	YM14	FT/SEC	
YM2(I)	YM21	FT/SEC	Second estimate in Runge-Kutta Integration.
	YM22	FT	
	YM23	FT	
	YM24	FT/SEC	
YM3(I)	YM31	FT/SEC	Third estimate in Runge-Kutta Integration.
	YM32	FT	
	YM33	FT	
	YM34	FT/SEC	
YM4(I)	YM41	FT/SEC	Fourth estimate in Runge-Kutta Integration.
	YM42	FT	
	YM43	FT	
	YM44	FT/SEC	
YT(I)	YT1	FT/SEC	Updated estimate in Runge-Kutta integration, used to find associated first derivatives.
	YT2	FT	
	YT3	FT	
	YT4	FT/SEC	
YV		FT	Y distance from van's axis to target center.
YY(I)	YY1	FT/SEC	Bomb's simulated position and velocity components. I=1 horizontal velocity I=2 vertical position I=3 horizontal position I=4 vertical velocity
	YY2	FT	
	YY3	FT	
	YY4	FT/SEC	
ZAC		FT	Aircraft's Z position component at release, from target center.
ZDOT		FT/SEC	Aircraft's Z velocity component at release.
ZV		FT	Z distance from van's axis to target center.

**** PHASE I ****

K= 1	R	THETA	PHI
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K= 2	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K= 3	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K= 4	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K= 5	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K= 6	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K= 7	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K= 8	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K= 9	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K=10	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K=11	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	820.31641	-0.001357	-0.000771
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K=12	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	820.31641	-0.001357	-0.000771
	910.62891	0.001243	-0.002979
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K=13	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	820.31641	-0.001357	-0.000771
	910.62891	0.001243	-0.002979
	983.69922	-0.000715	-0.000561
	0.0	0.0	0.0
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K=14	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	820.31641	-0.001357	-0.000771
	910.62891	0.001243	-0.002979
	983.69922	-0.000715	-0.000561
	1111.40625	-0.000043	-0.001038
	0.0	0.0	0.0
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K=15	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	820.31641	-0.001357	-0.000771
	910.62891	0.001243	-0.002979
	983.69922	-0.000715	-0.000561
	1111.40625	-0.000043	-0.001038
	1188.89453	-0.001440	-0.000165
	0.0	0.0	0.0
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K=16	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	820.31641	-0.001357	-0.000771
	910.62891	0.001243	-0.002979
	983.69922	-0.000715	-0.000561
	1111.40625	-0.000043	-0.001038
	1188.89453	-0.001440	-0.000165
	1254.76563	-0.002555	-0.002031
BAR=	10404.0547	0.126394	0.661351
TILDA=	10404.0547	0.126394	0.661351

K= 1	R	THETA	PHI
	0.0	0.0	0.0
	64.47656	-0.000058	-0.000074
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	820.31641	-0.001357	-0.000771
	910.62891	0.001243	-0.002979
	983.69922	-0.000715	-0.000561
	1111.40625	-0.000043	-0.001038
	1188.89453	-0.001440	-0.000165
	1254.76563	-0.002555	-0.002031
BAR=	10404.0547	0.126394	0.661351
TILDA=	9078.7773	0.129088	0.663162

K= 2	R	THETA	PHI
	0.0	0.0	0.0
	99.71484	0.001104	-0.000196
	148.62109	0.000612	-0.001642
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000248
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	820.31641	-0.001357	-0.000771
	910.62891	0.001243	-0.002979
	983.69922	-0.000715	-0.000561
	1111.40625	-0.000043	-0.001038
	1188.89453	-0.001440	-0.000165
	1254.76563	-0.002555	-0.002031
BAR=	10404.0547	0.126394	0.661351
TILDA=	9078.7773	0.129088	0.663162

K= 3	R	THETA	PHI
	0.0	0.0	0.0
	99.71484	0.001104	-0.000196
	174.91797	0.000860	-0.000382
	240.33594	-0.000373	-0.000076
	313.39844	-0.000952	-0.000438
	413.70313	0.000812	-0.000646
	517.37891	0.002230	-0.000354
	564.85938	-0.000309	-0.000246
	668.91016	0.000487	-0.002323
	752.15234	-0.001286	-0.001177
	820.31641	-0.001357	-0.000771
	910.62891	0.001243	-0.002979
	983.69922	-0.000715	-0.000561
	1111.40625	-0.000043	-0.001038
	1188.89453	-0.001440	-0.000165
	1254.76563	-0.002555	-0.002031
BAR=	10404.0547	0.126394	0.661351
TILDA=	9078.7773	0.129088	0.663162

**** PHASE II ****

RDOT= -83.0229 TDOT= 0.0004 PDOT= 0.0001

XDOT= -54.6771 YDOT= -656.3359 ZDOT= -506.4180

**** PHASE III ****

BALLASTICS FOR MK 83 MECH FUZE

CASE # 1

CHARACTERISTICS OF FLIGHT:
EJECTION VELOCITY (FT/SEC) - -4.8
DIVE ANGLE (RADIANS) - -0.7
VELOCITY OF AIRCRAFT (FPS) - 830.8
ALTITUDE OF AIRCRAFT (FT) - 5479.5
(USING A STEP SIZE OF 3.00 SECONDS)

TIME(SEC)	VERTICAL COMPONENTS			HORIZONTAL COMPONENTS		
	HEIGHT	DOWN VEL	DOWN ACC	RANGE	VELOCITY	ACCEL
3.00	3808.05	-603.94	-31.115	1962.85	652.83	-1.010
6.00	1856.66	-696.81	-30.755	3916.58	649.56	-1.207
9.00	-371.23	-788.05	-29.948	5859.23	645.29	-1.721
8.50	18.96	-773.03	-30.130	5536.44	646.12	-1.604
11.00	-2006.86	-847.12	-29.065	7146.15	641.33	-2.259
8.52	0.86	-773.73	-30.123	5551.55	646.08	-1.609
8.55	-17.25	-774.44	-30.115	5566.67	646.04	-1.615
8.52	0.00	-773.76	-30.122	5552.27	646.08	-1.610

IMPACT

ANGLE (DEGREES)	VELOCITY (FT/SEC)	HEIGHT (FT)	RANGE (FT)	TIME (SEC)
-50.14	1008.030	0.00	5552.27	8.52

**** PHASE IV ****

HIT COORDINATES

IN RELATION TO TARGETS RUN-IN-LINE		IN RELATION TO A/CS RUN-IN-LINE	
X	Y	X	Y
-440.8518	-1421.8164	-321.2920	-1453.5071

PILOTS CONDITIONS

DIVE ANGLE(DEGREES)	VELOCITY(KNOTS)	ALTITUDE(FT)
-37.557	492.2373	5479.4688

```
// EXEC FORTCLG,REGION.GO=100K
//SYSUT1 DD DATA
C   THE FOLLOWING PROGRAM IS A FORTRAN IMPLEMENTATION OF THE NO-DROP BOMB
C      SIMULATION. THIS, THE MAIN PROGRAM, STARTS IN THE TRACKING PHASE
C      OF THE PROBLEM AND TERMINATES WITH THE BOMB IMPACT PHASE.
C
C   THE READ STATEMENTS SIMULATE THE INPUT FROM THE RADAR TRACKING THE
C      AIRCRACT TO BOMB RELEASE (PICKLE). SAMPLE DATA READ IN, IS IN
C      TERMS OF X,Y, AND Z. THE SIX STATEMENTS FOLLOWING BOTH READS,
C      LABELED 'REMOVE', CONVERT DATA TO R, THETA, AND PHI.
C
C      R     -SLANT RANGE TO A/C              (FT)       STORED IN TRK(1,J)
C      THETA-ROTATION DIRECTION TO A/C        (RADIANS)     "    "  TRK(2,J)
C      PHI   -ELEVATION TO A/C                (RADIANS)     "    "  TRK(3,J)
C          (WHERE I RANGES FROM 1-16. ONLY THE 16 MOST RECENT POSITIONS
C            ARE STORED IN THESE WRAP-AROUND REGISTERS. THE ACTUAL VALUES
C            STORED IN TRK(I,J)= BASE VALUE - INPUT, WHERE THE BASE
C            VALUES = ACTUAL INPUT @ TRK(I,1))
C
C   DEFINITION OF VARIABLES USED IN MAIN PROGRAM
C      K      - J LOCATION IN TRK(I,J) AT PICKLE
C      BAR    - PREVIOUS BASE VALUE: USED FOR TRK(I,K+1) TO TRK(I,16)    FT
C      TILDA  - CURRENT BASE VALUE: USED FOR TRK(I,1) TO TRK(I,K)        FT
C      IPICKL- PICKLE SIGNAL
C      T      - RADAR OUTPUT SAMPLING TIME INTERVAL                    1/SEC
C      XDOT, YDOT, ZDOT     - A/C X,Y,Z VELOCITY COMPONENTS AT PICKLE   FPS
C      POS(1),POS(2),POS(3)- A/C R, THETA, AND PHI POSITION COMPONENTS AT
C                            PICKLE W/ RESPECT TO RADAR CENTER          FT
C      XV,YV,ZV    - RADAR X,Y,Z, DISPLACEMENT FROM TARGET CENTER       FT
C      XAC,YAC,ZAC- A/C X,Y,Z POSITION COMPONENTS AT PICKLE W/ RESPECT
C                   TO TARGET CENTER                                     FT
C      XWIND,YWIND- X,Y WIND COMPONENTS                                 FPS
C      GAMMA1- RUN-IN-LINE OF A/C W/ RESPECT TO THE FIXED TARGET'S
C              RUN-IN-LINE (RIL)                                    RADIANS
C      GAMMA - ADJUSTED A/C RIL (WIND CORRECTION)                   RADIANS
C      VTY   - TOTAL VEL OF A/C AT PICKLE                               FPS
C      ALPHA - DIVE ANGLE OF A/C AT PICKLE                          RADIANS
C      XIX,XIY    -IMPACT COORDINATES WRT TARGET RIL                    FT
C      XIXAC,XIYAC-   "          "       "   A/C   "                    FT
C
C
      COMMON /MNST/XV,YV,ZV,T,XWIND,YWIND
      COMMON /TRKSTO/TRK(3,16),BAR(3),TILDA(3),POS(3),VEL(3)
      DIMENSION S(3)
      CALL INPUT
      IPICKL=0
      DO 10 J=1,16
      DO 10 I=1,3
```

```
10      TRK(I,J)=0.
        I=0
        K=I+1
C
CC      TRACK AIRCRAFT (PHASE I)
C
        WRITE (6,100)
        READ(5,150) (TRK(L,K),L=1,3),IPICKL
        R1=SQRT(TRK(1,K)**2+TRK(2,K)**2+TRK(3,K)**2)                    REMOVE
        T1=ATAN(TRK(1,K)/TRK(2,K))                                      REMOVE
        P1=ATAN(TRK(3,K)/SQRT(TRK(1,K)**2+TRK(2,K)**2))                 REMOVE
        TRK(1,K)=R1                                                     REMOVE
        TRK(2,K)=T1                                                     REMOVE
        TRK(3,K)=P1                                                     REMOVE
        DO 1 L=1,3
        BAR(L)=TRK(L,K)
1       TILDA(L)=TRK(L,K)
2       DO 3 L=1,3
3       TRK(L,K)=TILDA(L)-TRK(L,K)
        WRITE(6,520) K
        WRITE (6,530) ((TRK(L,J),L=1,3),J=1,16)
        WRITE (6,540) (BAR(L), L=1,3)
        WRITE(6,550) (TILDA(L),L=1,3)
        IF (IPICKL.EQ.1) GO TO 7
        I=I+1
        I=MOD(I,16)
        K=I+1
        READ(5,150) (TRK(L,K),L=1,3),IPICKL
        R1=SQRT(TRK(1,K)**2+TRK(2,K)**2+TRK(3,K)**2)                    REMOVE
        T1=ATAN(TRK(1,K)/TRK(2,K))                                      REMOVE
        P1=ATAN(TRK(3,K)/SQRT(TRK(1,K)**2+TRK(2,K)**2))                 REMOVE
        TRK(1,K)=R1                                                     REMOVE
        TRK(2,K)=T1                                                     REMOVE
        TRK(3,K)=P1                                                     REMOVE
        IF(I.NE.0) GO TO 2
C
CC      UPDATE BASE VALUES
C
        DO 5 L=1,3
        BAR(L)=TILDA(L)
5       TILDA(L)=TRK(L,K)
        GO TO 2
C
CC      PERFORM VELOCITY FIT (PHASE II)
C
7       WRITE (6,110)
        CALL VELFIT(K,XDOT,YDOT,ZDOT)
        XDOT=XDOT*T
```

```
      YDOT=YDOT*T
      ZDOT=ZDOT*T
      WRITE(6,510) XDOT,YDOT,ZDOT
      XAC=POS(1)*SIN(POS(2))*COS(POS(3))+XV
      YAC=POS(1)*COS(POS(2))*COS(POS(3))+YV
      ZAC=POS(1)*SIN(POS(3))+ZV
      GAMMA1=ATAN(XDOT/YDOT)
      XDOT=XDOT+XWIND
      YDOT=YDOT+YWIND
      GAMMA=ATAN(XDOT/YDOT)
      VTY=SQRT(XDOT**2+YDOT**2+ZDOT**2)
      XX=SQRT(XDOT**2+YDOT**2)
      ALPHA=ATAN(ZDOT/XX)
C
CC    PERFORM BOMB DROP SIMULATION (PHASE III)
C
      WRITE (6,120)
      CALL TRAJ(ZAC,VTY,ALPHA,DR,TIME)
C
CC    CALCULATE IMPACT COORDINATES (PHASE IV)
C
      WRITE (6,130)
      XIX=DR*SIN(GAMMA)-XAC-XWIND*TIME
      XIY=DR*COS(GAMMA)-YAC-YWIND*TIME
      XIXAC=XIX*COS(GAMMA1)-XIY*SIN(GAMMA1)
      XIYAC=XIX*SIN(GAMMA1)+XIY*COS(GAMMA1)
      ALPHA=ALPHA*57.2957
      VTY=VTY/1.6878
C
CC    OUTPUT NECESSARY INFO TO PILOT
C
      WRITE(6,160) XIX,XIY,XIXAC,XIYAC,ALPHA,VTY,ZAC
100   FORMAT(//,1X,'**** PHASE I ****',//)
110   FORMAT(//,1X,'**** PHASE II ****',//)
120   FORMAT(//,1X,'**** PHASE III ****',//)
130   FORMAT(//,1X,'**** PHASE IV ****',//)
150   FORMAT (3F10.3,I1)
160   FORMAT(///,30X,'HIT COORDINATES',//,1X,'IN RELATION TO TARGETS  RU
     @N-IN-LINE',5X,'IN RELATION TO A/CS  RUN-IN-LINE',/,10X,'X',14X,'Y'
     @,23X,'X',13X,'Y',/,5X,F10.4,5X,F10.4,14X,F10.4,4X,F10.4,////,22X,'
     @PILOTS  CONDITIONS',//,6X,'DIVE ANGLE(DEGREES)',5X,'VELOCITY(KNOTS)
     @',6X,'ALTITUDE(FT)',/,11X,F7.3,12X,F10.4,11X,F10.4)
510   FORMAT(//,1X,'XDOT=',F10.4,5X,'YDOT=',F10.4,5X,'ZDOT=',F10.4)
520   FORMAT (//,1X,'K=',I2,8X,'R',8X,'THETA',7X,'PHI')
530   FORMAT (7X,F10.5,1X,F10.6,1X,F10.6)
540   FORMAT (1X,'BAR=',2X,F10.4,1X,F10.6,1X,F10.6)
550   FORMAT (1X,'TILDA=',F10.4,1X,F10.6,1X,F10.6)
      STOP
```

```
      END
C  INPUT SUBROUTINE - VARIABLES HERE ARE CONSTANT THROUGHOUT THE PROGRAM
C
C     *INPUTED TABLES*
C      ALL TABLES ARE INPUTED AS DIVIDED DIFFERENCE TABLES OF THE FORM
C                                                        VALUE(I+1)-VALUE(I)
C          ENTRY(I),VALUE(I),DIFF(I)  WHERE DIFF(I)=--------------------
C                                                        ENTRY(I+1)-ENTRY(I)
C    TABLES USED
C      MACH VS DRAG              - AMACH(I)  ACOEF(I)   DCD(I)
C      ALTITUDE VS DENSITY       -   H(I)     RHOT(I)  DRHO(I)  SLUGS/CU FT
C      ALT VS 1/SPEED OF SOUND-    H(I)      VSBT(I)  DVSB(I)         1/FPS
C        LKCD,LKRHO,LKVSB REFER TO PREVIOUS I VALUE IN RESULTANT
C         VALUE(I) FOR SPEED IN TABLE LOOK-UP
C      *TABLES FOR VELOCITY FIT ROUTINE*
C       C(I,J) LEGENDRE DISCRET POYNOMIALS FOR J PCINT FIT   I=0 FOR A0
C                                                             =1 FOR A1
C                                                             =2 FOR A2
C       SO(I)  1/SUMMATION OF O(I,J) J=1,16
C
C      *CTHER VARIABLES*
C       CAT  - TARGET ALTITUDE                                    FT
C       DEG  - RADIANS TO DEGREES CONVERSION CONSTANT
C       RAD  - DEGREES TO RADIANS CONVERSION CONSTANT
C       DTO  - TIME STEP FOR SIMULATION ROUTINE (TRAJ)          SEC
C       DELT1- FIN OPENING PERPENDICULAR TO FLIGHT PATH         SEC
C       DELT2-  "      "       PARALLEL     "    "    "       SEC
C       VE2B - EJECTION VEL      "          "    "    "       FPS
C       VE3B -     "      " PERPENDICULAR "    "    "       FPS
C       IROCK-   =0  SINGLE DRAG BOMB
C                =1  DUAL DRAG BOMB
C       XV,YV,ZV    - SEE MAIN COMMENTS
C       XWIND,YWIND-   "   "       "
C          T        -   "   "       "
C       S    - CROSS SECTIONAL AREA OF BOMB =.7854*DIAMETER**2    FT**2
C       MASS- 1/MASS OF BOMB                =32.174/WEIGHT         1/LB
C
      SUBROUTINE INPUT
      COMMON /MNST/XV,YV,ZV,T,XWIND,YWIND
      COMMON /TKST/ O(3,16),SO(3)
      COMMON /TABSO/MMACH(2),MH,AMACH(200,2),H(61,1),ACOEF(200,2),RHOT(6
     @1,1),VSBT(61,1),LKCD,LKRHO,LKVSB,DRHO(61,1),DVSB(61,1),DCD(200,2)
      COMMON /TJST/DTO,DEG,RAD,VE2B,VE3B,DELT1,DELT2,CAT,IROCK
      COMMON /DESTO/MASS(2),S(2)
      REAL*4 MASS
      DTO=3.
      DEG=57.30
```

```
      RAD=.01745
      LKCD=1
      LKRHO=1
      LKVSB=1
      XV = 0.
      YV = 0.
      ZV = 0.
      XWIND=0.
      YWIND=0.
      T=10.
      CAT = 0.
      VE2B = 0.
      VE3B = -4.8
      DELT1 = 0.
      DELT2 = 0.
      MH=61
      READ(5,150) ((O(I,J),I=1,3),J=1,16)
      READ(5,160) (SO(I),I=1,3)
      READ(5,140)(H(I,1),RHOT(I,1),DRHO(I,1),VSBT(I,1),DVSB(I,1),I=1,61)
      READ (5,100) IROCK
      DO 10 L=1,IROCK
      READ(5,110) MMACH(L)
      K = MMACH(L)
      READ (5,120) (AMACH(I,L),ACOEF(I,L),DCD(I,L),I=1,K)
10    READ (5,130) MASS(L),S(L)
      IROCK = IROCK - 1
100   FORMAT(I1)
110   FORMAT(I3)
120   FORMAT(F8.5,1X,F8.5,1X,F10.8)
130   FORMAT(F10.8,1X,F11.6)
140   FORMAT(2X,F6.0,7X,F10.8,4X,F11.9,5X,F10.8,4X,F11.9)
150   FORMAT(1X,3F10.7)
160   FORMAT(1X,3F10.6)
      RETURN
      END
C   VELOCITY FIT SUBROUTINE
C      INPUT- REGISTERS CONTAINING R, THETA, PHI AND K RELEASE POSITION
C
C                 ---------------------------------
C      TRK(1,J)  | | | | | | | | | | | | | | | | | R
C                 ---------------------------------
C      TRK(2,J)  | | | | | | | | | | | | | | | | | THETA
C                 ---------------------------------
C      TRK(3,J)  | | | | | | | | | | | | | | | | | PHI
C                 ---------------------------------
C                   MOST RECENT  K  LEAST RECENT
C                                |
C                             PICKLE
```

```
C
CCC     USING LEGENDRE POLYNOMIALS (SECOND ORDER)
CCC         A0,A1,A2 (A(I) I=1,3) ARE CALULATED TO FIND POS,VEL FOR R
CCC                                                                THETA
CCC                                                                PHI
CCC     THE O(I,J) MUST BE MATCHED TO THE PROPER TRK(I,J) FROM LEAST
CCC           RECENT TO MOST RECENT IN THAT ORDER.
CCC
C
      SUBROUTINE VELFIT(K,XDOT,YDOT,ZDOT)
      COMMON /TRKSTO/TRK(3,16),BAR(3),TILDA(3),POS(3),VEL(3)
      COMMON /TKST/ O(3,16),SO(3)
      DIMENSION A(3)
C
CCC   FOR R, THETA, AND PHI
C
      DO 4 L=1,3
C
CCC   FOR A0, A1, AND A2
C
      DO 3 I=1,3
      AA=0.
      AAA=0.
      S2=0.
      S1=0.
      A(I)=0.
      K1=16-K
      IF(K1.EQ.0) GO TO 7
C
CCC   SUM THE LEAST RECENT POSITIONS AND MULTIPLY BY BASE VALUE
C
      DO 5 J=1,K1
      AA=AA+O(I,J)
      K2=K+J
5     S1=S1+O(I,J)*TRK(L,K2)
      AA=AA*BAR(L)
C
CCC   ENTRY FOR WHEN PICKLE OCCURS AT END OF REGISTER (K=16)
CCC   SUM MUST RECENT POSITIONS AND MULTIPLY BY BASE VALUE
CC
7     K1=K1+1
      DO 6 J=K1,16
6     AAA=AAA+O(I,J)
      AAA=AAA*TILDA(L)
      DO 2 J=1,K
      K2=K1+J-1
2     S2=S2+O(I,K2)*TRK(L,J)
C
```

```
C      SOLVE FOR A0,A1, AND A2
C
3      A(I)=(AA+AAA-S1-S2)*SO(I)
C
C      SOLVE FOR POSITIONS AND VELOCITIES (R,THETA,PHI) ON FITTED CURVE
C       AT PICKLE
C
       POS(L)=A(1)-A(2)+A(3)
4      VEL(L)=(-(2.*A(2))/15.)+((6.*A(3))/14.)
       WRITE (6,500) (VEL(I), I=1,3)
C
C      CONVERT TO X,Y, AND Z VELOCITIES
C
       XDOT=SIN(POS(2))*COS(POS(3))*VEL(1)+POS(1)*COS(POS(2))*COS(POS(3))
      @*VEL(2)-POS(1)*SIN(POS(2))*SIN(POS(3))*VEL(3)
       YDOT=COS(POS(2))*COS(POS(3))*VEL(1)-POS(1)*SIN(POS(2))*COS(POS(3))
      @*VEL(2)-POS(1)*COS(POS(2))*SIN(POS(3))*VEL(3)
       ZDOT=SIN(POS(3))*VEL(1)+POS(1)*COS(POS(3))*VEL(3)
500    FORMAT(/,1X,'RDOT=',F10.4,5X,'TDOT=',F10.4,5X,'PDOT=',F10.4)
       RETURN
       END
C   TRAJ SUBROUTINE
C     *DEFINITION OF VARIABLES*
C       YY(1)=SY(1)- HORIZONTAL VELOCITY  DY(1)- HORIZONTAL ACCELERATION
C       YY(2)=SY(2)-  VERTICAL  POSITION  DY(2)-  VERTICAL  VELOCITY
C       YY(3)=SY(3)- HORIZONTAL POSITION  DY(3)- HORIZONTAL VELOCITY
C       YY(4)=SY(4)-  VERTICAL  VELOCITY  DY(4)  VERTICAL  ACCELERATION
C       DELT - TOTAL FIN OPENING TIME                                  SEC
C       GAMMA- RELEASE ANGLE OF THE BOMB (ADJUSTED FOR EJECTION VEL)  RADS
C       VEL  - INITIAL VELOCITY OF THE BOMB                            FPS
C       JROCK-  =1  SINGLE DRAG OR FIRST STAGE OF DUAL DRAG BOMB
C               =2  SECOND STAGE OF DUAL DRAG BOMB
C       DT   - TIME STEP                                               SEC
C       TIME - TOTAL TIME OF FALL                                      SEC
C       KGATE- FOR DUAL DRAG BOMB ONLY
C              =1   FIRST STAGE (BOMB @ RELEASE)
C              =2   FIRST STAGE UP TO FIN OPENING TIME
C              =3   AT FIN OPENING TIME
C              =4   SECOND STAGE
C       NMACH- TOTAL # OF ENTRIES IN CURRENT DRAG TABLE
C            (NOTE: FOR DUAL DRAG BOMBS, THERE ARE 2 MACH VS DRAG TABLES)
C
C
       SUBROUTINE TRAJ(Y,V,GAM,XT,TF)
       COMMON /TJST/DTO,DEG,RAD,VE2B,VE3B,DELT1,DELT2,CAT,IROCK
       COMMON /TABSO/MMACH(2),MH,AMACH(200,2),H(61,1),ACOEF(200,2),RHOT(6
      @1,1),VSBT(61,1),LKCD,LKRHO,LKVSB,DRHO(61,1),DVSB(61,1),DCD(200,2)
       COMMON /TABSTO/TIME
```

```
      DIMENSION YY(4),DY(4),SY(4)
      WRITE (6,240)
      II=1
      WRITE (6,250) II,VE3B,GAM,V,Y,DTO
      DT = DTO
C
CC    INITALIZE VELOCITY AND RELEASE ANGLE
C
      GAMMA = GAM + ATAN2 (VE3B,V+VE2B)
      VEL = SQRT((V+VE2B)**2 + VE3B**2)
      IF (DELT1 .NE. 0) DELT = DELT1 - DELT2 * VEL
      YY(4) = VEL * SIN(GAMMA)
      YY(1) = VEL * COS(GAMMA)
      YY(2) = Y
      YY(3) = 0.
      SY(4) = YY(4)
      TIME = 0.
      NMACH = MMACH (1)
      JROCK = 1
      WRITE (6,200)
      IF (IROCK .GT. 0) GO TO 5
C
CC    SINGLE DRAG BOMB
C
10    CALL EINT (TIME,YY,DT,JROCK)
      GO TO 50
C
CC    DUAL DRAG BOMB
C
 5    KGATE = 1
      IF (DELT .LT. DT) DT = DELT
      IF (DELT .LE. 0) KGATE = 4
15    CALL EINT (TIME,YY,DT,JROCK)
      GO TO (25,45,45,50),KGATE
25    IF (TIME + DT - DELT) 50,35,55
35    KGATE = 3
      GO TO 50
45    IF (DELT - TIME .LT. .001) GO TO 65
55    KGATE = 2
      DT = DELT - TIME
      IF (DT .GE. .0001) GO TO 50
65    KGATE = 4
      DT = DTO
      JROCK = 2
      NMACH = MMACH(2)
C
CC    CHECK FOR IMPACT
C
```

```
50      IF (ABS(YY(2) - CAT) .LT. .1) GO TO 80
        IF (YY(2) - CAT) 70,80,60
C
CC      SAVE YY VALUES FOR POSSIBLE INTERPOLATION
CC
60      DO 2 IB =1,4
2       SY(IB) = YY(IB)
        ST = TIME
        IF (NMACH .EQ. 0 .OR. IROCK .GT. 0) GO TO 15
        GO TO 10
C
CC      INTERPOLATE: BOMB ALTITUDE < TARGET ALTITUDE
CC
70      DT =((SY(2)-CAT)*(ST-TIME))/(YY(2)-SY(2))
        TIME = ST
        DO 3 IB=1,4
 3      YY(IB) = SY(IB)
        IF (NMACH .EQ. 0 .OR. IROCK .GT. 0) GO TO 15
        GO TO 10
C
CC      IMPACT
CC
80      GT = ATAN2(YY(4),YY(1)) * DEG
        VT = SQRT (YY(1)**2 + YY(4)**2)
        YT = YY(2)
        XT = YY(3)
        TF = TIME
        WRITE (6,220)
        WRITE (6,230) GT,VT,YT,XT,TF
200     FORMAT('0',1X,'TIME(SEC)',7X,'VERTICAL COMPONENTS',20X,'HORIZONTAL
       # COMPONENTS',16X,//,12X,'HEIGHT',4X,'DOWN VEL',4X,'DOWN ACC',10X,'
       #RANGE',4X,'VELOCITY',6X,'ACCEL',//)
220     FORMAT ('0',30X,'IMPACT',//,1X,'ANGLE (DEGREES)',3X,'VELOCITY (FT/
       #SEC)',4X,'HEIGHT (FT)',2X,'RANGE (FT)',3X,'TIME (SEC)',//)
230     FORMAT (3X,F7.2,12X,F9.3,11X,F6.2,6X,F8.2,5X,F7.2,//////)
240     FORMAT(40X,'BALLASTICS FOR MK 83 MECH FUZE',///)
250     FORMAT(10X,'CASE # ',I2,//,1X,'CHARACTERISTICS OF FLIGHT:',/,6X,'E
       #JECTION VELOCITY (FT/SEC)  - ',F5.1,/,6X,'DIVE ANGLE (RADIANS)',8X
       #,'- ',F5.1,/,6X,'VELOCITY OF AIRCRAFT (FPS)  - ',F8.1,/,6X,'ALTITU
       #DE OF AIRCRAFT (FT)   - ',F7.1,/,1X,'(USING A STEP SIZE OF ',F4.2,
       #' SECONDS)',///)
        RETURN
        END
C   EINT SUBROUTINE
CC
CC      A FOURTH ORDER RUNGA KUTTA INTEGRATION ROUTINE TO UPDATE YY
CC          VALUES ON THE BASIS OF THEIR FIRST DERIVATES
C
```

```
C      DELTA- TIME INTERVAL     SEC
CC     L     - BOMB STAGE
CC
CC    *FORM OF THE ROUTINE*
CC       FNA=DY/DT
CC       M1=DELTA*FNA(YY)
CC       M2=DELTA*FNA(YY+M1/2)
CC       M3=DELTA*FNA(YY+M2/2)
CC       M4=DELTA*FNA(YY+M3)
CC          THUS UPDATING
CC       YY=YY+(M1+2*M2+2*M3+M4)/6
CC       TIME=TIME+DELTA
CC
C
      SUBROUTINE EINT (TIME,YY,DELTA,L)
      DIMENSION YY(4),DY(4),YT(4),YM1(4),YM2(4),YM3(4),YM4(4)
      CALL DERIV (YY,L,DY)
      DO 1 I=1,4
      YM1(I) = DELTA * DY(I)
1     YT(I) = YY(I) + YM1(I)/2
      CALL DERIV (YT,L,DY)
      DO 2 I=1,4
      YM2(I) = DELTA * DY(I)
2     YT(I) = YY(I) + YM2(I)/2
      CALL DERIV (YT,L,DY)
      DO 3 I=1,4
      YM3(I) = DELTA * DY(I)
3     YT(I) = YY(I) + YM3(I)
      CALL DERIV (YT,L,DY)
      DO 4 I=1,4
      YM4(I) = DELTA * DY(I)
4     YY(I) = YY(I)+(YM1(I)+2*YM2(I)+2*YM3(I)+YM4(I))/6
      TIME = TIME + DELTA
      WRITE (6,210) TIME,YY(2),YY(4),DY(4),YY(3),YY(1),DY(1)
210   FORMAT(1X,F7.2,3X,F8.2,3X,F8.2,3X,F9.3,6X,F8.2,3X,F8.2,3X,F9.3)
5     RETURN
      END
C  DERIV SUBROUTINE
C
C     A DERIVATIVE ROUTINE TO FIND THE FIRST DERIVATES OF THE YY
C      COMPONENTS
C
C     DEN - A FUNCTION OF DRAG, CROSS-SECTIONAL AREA AND MASS OF BOMB
C           USED, DENSITY AND VELOCITY AT BOMB ALTITUDE: USED TO
C           DETERMINE HORIZONTAL AND VERTICAL ACCELERATION DY(1) DY(4)
C     VERTICAL ACCELERATION ALSO TAKES INTO ACCOUNT THE EARTH'S
C           GRAVITATIONAL PULL = 32.05 FT/SEC**2
C     QS - INTERMEDIATE STORAGE AREA
```

```
C
C
      SUBROUTINE DERIV (YY,L,DY)
      COMMON /DESTO/MASS(2),S(2)
      COMMON /TABSTO/MMACH(2),MH,AMACH(200,2),H(61,1),ACOEF(200,2),RHOT(6
     @1,1),VSBT(61,1),LKCD,LKRHO,LKVSB,DRHO(61,1),DVSB(61,1),DCD(200,2)
      DIMENSION YY(4),DY(4)
      REAL*4 MASS,MACH
      DY(2) = YY(4)
      DY(3) = YY(1)
      VEL = SQRT (DY(2)**2 + DY(3)**2)
C
CC    FIND DENSITY OF AIR AT BOMB ALTITUDE
C
      CALL TABLE(YY(2),1,RHO,LKRHO,H,RHOT,DRHO,MH)
C
CC    FIND 1/SPEED OF SOUND TO CALCULATE MACH NUMBER AT BOMB ALTITUDE
C
      CALL TABLE(YY(2),1,VSB,LKVSB,H,VSBT,DVSB,MH)
      QS=RHO*VEL*S(L)
      MACH=VEL*VSB
      MM=MMACH(L)
C
CC    FIND BOMB'S DRAG COEFFICIENT AT PRESENT MACH NUMBER
C
      CALL TABLE(MACH,L,CD,LKCD,AMACH,ACOEF,DCD,MM)
      DEN=-CD*QS*MASS(L)
      DY(1) = DEN*YY(1)
      DY(4) = DEN*YY(4) -32.05
      RETURN
      END
C  TABLE SUBROUTINE
C
CCC   PRINCIPLE IDEA AND DEFINITION OF VARIABLES IN INPUT COMMENTS
CC
      SUBROUTINE TABLE(ENTRY,NUM,VALUE,LOOK,ENT,VAL,DIFF,MAX)
      COMMON /TABSTO/TIME
      DIMENSION ENT(MAX,NUM),VAL(MAX,NUM),DIFF(MAX,NUM)
      IF (ENTRY-ENT(LOOK,NUM)) 1,10,5
 1    IF (LOOK.LE.1) GO TO 10
      LOOK=LOOK-1
      IF (ENTRY-ENT(LOOK,NUM)) 1,10,4
 4    K=LOOK
      GO TO 9
 5    IF (LOOK.GE.MAX) GO TO 10
      LOOK=LOOK+1
      IF (ENTRY-ENT(LOOK,NUM)) 8,10,5
```

```
 8      K=LOOK-1
 9      VALUE=VAL(K,NUM)+(ENTRY-ENT(K,NUM))*DIFF(K,NUM)
        GO TO 11
10      VALUE=VAL(LOOK,NUM)
11      CONTINUE
        RETURN
        END
//FORT.SYSIN DD *
```

```
//* 8008 PLM COMPILE ONLY
//PASS1   EXEC  PGM=PLM1,REGION=100K
//STEPLIB DD UNIT=2314,VOL=SER=LINDA,DISP=SHR,DSN=C0012.MCS
//FTC6FC01 DD DUMMY
//PRINTER DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=726)
//OUTPUT6 DD UNIT=SYSDA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//         DISP=(,PASS),SPACE=(TRK,(4,2))
//OUTPUT7 DD UNIT=SYSDA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//         DISP=(,PASS),SPACE=(TRK,(4,2))
//CARD DD *
$M
$S
DECLARE (VEL,ALPHA) ADDRESS,
        (T,GAMMA1,XX,GAMMA,IMPX,IMPY,IMPXAC,IMPYAC) ADDRESS,
        (DEG,FPS) ADDRESS,
         (H,RHOT,DRHO,VSBT,DVSB,AMACH,ACOEF,DCD) ADDRESS,
        (XWIND,YWIND,XVAN,YVAN,ZVAN) ADDRESS,
         (S,MASS) ADDRESS, BS BASED S BYTE, BMASS BASED MASS BYTE,
        (I,PICKL) BYTE INITIAL (0,0),
        (TRK1,TRK2,TRK3,BAR1,BAR2,BAR3,TILDA1,TILDA2,TILDA3) ADDRESS,
        (XDOT,YDOT,ZDOT,XAC,YAC,ZAC) ADDRESS,
         (DR,TIME) ADDRESS INITIAL (0,0),
        BTRK1 BASED TRK1 BYTE,
        BTRK2 BASED TRK2 BYTE,
        BTRK3 BASED TRK3 BYTE;


  /* FLOATING POINT ADD ROUTINE*/
ADD: PROCEDURE (XA,YA) ADDRESS;
     DECLARE (XA,YA,ZA,XX,YY,ZZ) ADDRESS,
             (I,X2,Y2,DIGIT,SINE) BYTE,
             X BASED XA BYTE,
             Y BASED YA BYTE,
             Z BASED ZA BYTE;

     SETXX: PROCEDURE;
            ZZ=XX-YY;
     END SETXX;

     SETYY: PROCEDURE;
            ZZ=YY-XX;
     END SETYY;

     GET$X: PROCEDURE (I) BYTE;
            DECLARE I BYTE;
            RETURN X(I);
     END GET$X;
```

```
GET$Y: PROCEDURE (I) BYTE;
       DECLARE I BYTE;
       RETURN Y(I);
END GET$Y;

GET$Z: PROCEDURE (I) BYTE;
       DECLARE I BYTE;
       RETURN Z(I);
END GET$Z;

/* PROCEDURE TO LEFT JUSTIFY MANTISSA IN BINARY */
 SHT$MANT: PROCEDURE BYTE;
     DECLARE (I,Z0,Z1) BYTE;
     Z0=HIGH(ZZ); Z1=LOW(ZZ);
     IF Z0<>0 THEN DO;
          IF Z0 AND 0F0H<>0 THEN DO;
               IF Z0 AND 0C0H<>0 THEN DO;
                    IF Z0 AND 80H<>0 THEN DO; I=0; RETURN I; END;
                    I=1; RETURN I;
                    END;
               IF Z0 AND 20H<>0 THEN DO; I=2; RETURN I; END;
               I=3; RETURN I;
               END;
          IF Z0 AND 0CH<>0 THEN DO;
               IF Z0 AND 08H<>0 THEN DO; I=4; RETURN I; END;
               I=5; RETURN I;
               END;
          IF Z0 AND 02H<>0 THEN DO; I=6; RETURN I; END;
          I=7; RETURN I;
          END;
     IF Z1 AND 0F0H<>0 THEN DO;
          IF Z1 AND 0C0H<>0 THEN DO;
               IF Z1 AND 80H<>0 THEN DO; I=8; RETURN I; END;
               I=9; RETURN I;
               END;
          IF Z1 AND 20H<>0 THEN DO; I=10; RETURN I; END;
          I=11; RETURN I;
          END;
     IF Z1 AND 0CH<>0 THEN DO;
          IF Z1 AND 08H<>0 THEN DO; I=12; RETURN I; END;
          I=13; RETURN I;
          END;
     IF Z1 AND 02H<>0 THEN DO; I=14; RETURN I; END;
     I=15; RETURN I;
 END SHT$MANT;

 ADJUST: PROCEDURE (I);
     DECLARE I BYTE;
```

```
        ZZ=SHL(ZZ,I);
        Z(2)=(Z(2) AND 0C0H)+I;
    END ADJUST;

    XX=SHL(DOUBLE(X),8) OR GET$X(1);
    YY=SHL(DOUBLE(Y),8) OR GET$Y(1);
    X2= GET$X(2) AND 7FH;
    Y2= GET$Y(2) AND 7FH;
    DIGIT=X2-Y2;
    SINE= GET$X(2) XOR GET$Y(2);
    IF SIGN(DIGIT) = 1 THEN DIGIT =-DIGIT;
    IF DIGIT >= 16  THEN GOTO SIG1;
     IF Y2 = X2 THEN DO;
/* EXPONENTS EQUAL */
        GET$Z(2)=GET$X(2);
         IF XX>YY THEN DO;
/* Y > X */
              IF SINE < 80H THEN GOTO ENT1;
              CALL SETYY;
              CALL ADJUST(SHT$MANT);
              GOTO ENT3;
              END;
        IF YY>XX THEN DO;
/* X > Y */
              IF SINE < 80H THEN GOTO ENT1;
              CALL SETXX;
              CALL ADJUST(SHT$MANT);
              GOTO ENT3;
              END;
        IF XX = YY THEN DO;
/* X = Y */
              IF SINE < 80H THEN GOTO ENT1;
              ZZ=0;
              GET$Z(2) = 0;
              GOTO ENT3;
              END;
        END;
    IF Y2 > X2 THEN DO;
/* Y > X */
        GET$Z(2) = GET$Y(2);
        XX=SHR(XX,DIGIT);
        IF SINE < 80H THEN GOTO ENT1;
        CALL SETYY;
        CALL ADJUST(SHT$MANT);
        GOTO ENT3;
        END;
    IF X2 > Y2 THEN DO;
/* X > Y */
```

```
            GET$Z(2) = GET$X(2);
            YY=SHR(YY,DIGIT);
            IF SINE < 80H THEN GOTO ENT1;
            CALL SETXX;
            CALL ADJUST(SHT$MANT);
            GOTO ENT3;
            END;
  /* VARIABLES NOT WITHIN SIGNIFICANCE RANGE */
SIG1: IF X2 > Y2 THEN RETURN XA; RETURN YA;
ENT1: ZZ=XX+YY;
ENT2: IF CARRY THEN DO;
            ZZ=SCR(ZZ,1);
            GET$Z(2) = GET$Z(2)+1;
            END;
ENT3: Z=HIGH(ZZ);
      GET$Z(1)=LOW(ZZ);
      RETURN ZA;
END ADD;

  /* FLOATING POINT MULTIPLY ROUTINE */
MULT: PROCEDURE (XA,YA) ADDRESS;
      DECLARE (X0,X1,X2,X3,Y0,Y1,Y2,Y3,FLAG1) BYTE,
               (T1,T2,T3,T4,T5,T6,T7,T8,CHECK) BYTE,
               (XA,YA,ZA) ADDRESS,
               X BASED XA BYTE,
               Y BASED YA BYTE,
               Z BASED ZA BYTE;

  /* TWO HEX DIGIT MILTIPLY TABLE */
      TABLE: PROCEDURE (I) BYTE;
          DECLARE I BYTE, X (256) BYTE INITIAL (
        00H,01H,02H,03H,04H,05H,06H,07H,08H,09H,0AH,0BH,0CH,0DH,0EH,0FH,
        00H,02H,04H,06H,08H,0AH,0CH,0EH,10H,12H,14H,16H,18H,1AH,1CH,1EH,
        00H,03H,06H,09H,0CH,0FH,12H,15H,18H,1BH,1EH,21H,24H,27H,2AH,2DH,
        00H,04H,08H,0CH,10H,14H,18H,1CH,20H,24H,28H,2CH,30H,34H,38H,3CH,
        00H,05H,0AH,0FH,14H,19H,1EH,23H,28H,2DH,32H,37H,3CH,41H,46H,4BH,
        00H,06H,0CH,12H,18H,1EH,24H,2AH,30H,36H,3CH,42H,48H,4EH,54H,5AH,
        00H,07H,0EH,15H,1CH,23H,2AH,31H,38H,3FH,46H,4DH,54H,5BH,62H,69H,
        00H,08H,10H,18H,20H,28H,30H,38H,40H,48H,50H,58H,60H,68H,70H,78H,
        00H,09H,12H,1BH,24H,2DH,36H,3FH,48H,51H,5AH,63H,6CH,75H,7EH,87H,
        00H,0AH,14H,1EH,28H,32H,3CH,46H,50H,5AH,64H,6EH,78H,82H,8CH,96H,
        00H,0BH,16H,21H,2CH,37H,42H,4DH,58H,63H,6EH,79H,84H,8FH,9AH,0A5H,
        00H,0CH,18H,24H,30H,3CH,48H,54H,60H,6CH,78H,84H,90H,9CH,0A8H,
                0B4H,
        00H,0DH,1AH,27H,34H,41H,4EH,5BH,68H,75H,82H,8FH,9CH,0A9H,0B6H,
                0C3H,
        00H,0EH,1CH,2AH,38H,46H,54H,62H,70H,7EH,8CH,9AH,0A8H,0B6H,0C4H,
                0D2H,
```

```
 C0H,0FH,1EH,2DH,3CH,4BH,5AH,69H,78H,87H,96H,0A5H,0B4H,0C3H,0D2H,
        0E1H);
  RETURN X(I);
END TABLE;

ADDFLAG: PROCEDURE;
         FLAG1=FLAG1+1;
END ADDFLAG;

SETZ0: PROCEDURE;
       Z=SCR(Z,1);
END SETZ0;

SETZ1: PROCEDURE;
       Z(1)=SHR(Z(1),1);
END SETZ1;

SETZ1C: PROCEDURE;
        Z(1)=SHR(Z(1),1) OR 80H;
END SETZ1C;

SETCHECK: PROCEDURE;
          CHECK=Z(0) AND 01H;
END SETCHECK;

GET$X: PROCEDURE (I) BYTE;
       DECLARE I BYTE;
       RETURN X(I);
END GET$X;

GET$Y: PROCEDURE (I) BYTE;
       DECLARE I BYTE;
       RETURN Y(I);
END GET$Y;

GET$Z: PROCEDURE (I) BYTE;
       DECLARE I BYTE;
       RETURN Z(I);
END GET$Z;

XC= GET$X(1) AND 0FH;
X1= SHR(GET$X(1),4);
X2=X AND 0FH;
X3=SHR(X,4);
YC= SHL(GET$Y(1),4);
Y1= GET$Y(1) AND 0F0H;
Y2= SHL(Y,4);
Y3=Y AND 0F0H;
```

```
      T2=SHR(TABLE(X1+Y1),4)+TABLE(X1+Y2);
      T3=SHR(T2,4)+TABLE(X1+Y3);
      T1=T1+(SHL(T3,4)+(T2 AND 0FH));
      IF CARRY THEN FLAG1=1; FLAG1=0;
      T4=SHR(TABLE(X2+Y0),4)+TABLE(X2+Y1);
      T5=SHR(T4,4)+TABLE(X2+Y2);
      T6=SHR(T5,4)+TABLE(X2+Y3);
      T7=TABLE(X3+Y0);
      T8=SHR(T7,4)+TABLE(X3+Y1);
      T4=((T7 AND 0FH)+SHL(T8,4))+((T4 AND 0FH)+SHL(T5,4));
      IF CARRY THEN CALL ADDFLAG;
      T1=T1+T4;
      IF CARRY THEN CALL ADDFLAG;
      T2=SHR(T3,4)+FLAG1;
      T5=SHR(T8,4)+TABLE(X3+Y2);
      T8=SHR(T5,4)+TABLE(X3+Y3);
      T6=T6+((T5 AND 0FH)+SHL(T8,4));
      IF CARRY THEN FLAG1=1; FLAG1=0;
      T2=T2+T6;
      IF CARRY THEN CALL ADDFLAG;
      T3=SHR(T8,4)+FLAG1;
      IF T3<8H THEN DO;
/* ZERO IN LEFT MOST MANTISSA BIT */
         FLAG1=1;
         Z =SHL(T3,5) OR SHR(T2,3);
         GET$Z(1)= SHL(T2,5) OR SHR(T1,3);
         IF (T1 AND 04H) <> 0 THEN DO;
 /* ROUND UP MANTISSA */
            GET$Z(1)= GET$Z(1)+1;
            IF CARRY THEN DO;
               Z=Z+1;
               IF CARRY THEN DO;
                  CALL SETCHECK;
                  IF CHECK = 0 THEN DO;
                     CALL SETZ0;
                     CALL SETZ1;
                     END;
                  IF CHECK = 1 THEN DO;
                     CALL SETZ0;
                     CALL SETZ1C;
                     END;
                  END;
               END;
            END;
      IF T3 >= 8H THEN DO;
 /* ONE IN LEFT MOST MANTISSA BIT */
         FLAG1=0;
```

```
            Z=SHL(T3,4) OR SHR(T2,4);
        GET$Z(1)=SHL(T2,4) OR SHR(T1,4);
            IF (T1 AND 08H) <>0 THEN DO;
    /* ROUND UP MANTISSA */
                GET$Z(1)=GET$Z(1)+1;
                IF CARRY THEN DO;
                    Z=Z+1;
                    IF CARRY THEN DO;
                        CALL SETCHECK;
                        IF CHECK = 0 THEN DO;
                            CALL SETZ0;
                            CALL SETZ1;
                            END;
                        IF CHECK = 1 THEN DO;
                            CALL SETZ0;
                            CALL SETZ1C;
                            END;
                        END;
                    END;
                END;
            END;
    /* SET PRODUCT EXPONENT */
        GET$Z(2)=(((GET$X(2) AND 80H) XOR (GET$Y(2) AND 80H)) OR
                  ((GET$X(2) AND 7FH) + (GET$Y(2) AND 7FH))) - FLAG1;
        RETURN ZA;
END MULT;

    /* FLOATING POINT DIVIDE ROUTINE */
DIV:  PROCEDURE (XA,YA) ADDRESS;
      DECLARE (XA,YA,ZA,XX,YY,ZZ,T) ADDRESS,
              I BYTE,
              X BASED XA BYTE,
              Y BASED YA BYTE,
              Z BASED ZA BYTE;

      SET$T: PROCEDURE;
             T=XX-YY;
      END SET$T;

      SHIFT$X: PROCEDURE;
               XX=SHL(XX,1);
      END SHIFT$X;

      GET$X: PROCEDURE (I) BYTE;
             DECLARE I BYTE;
             RETURN X(I);
      END GET$X;
```

```
        GET$Y: PROCEDURE (I) BYTE;
                DECLARE I BYTE;
                RETURN Y(I);
        END GET$Y;

        GET$Z: PROCEDURE (I) BYTE;
                DECLARE I BYTE;
                RETURN Z(I);
        END GET$Z;

        XX=SHL(DOUBLE(X),8) OR GET$X(1);
        YY=SHL(DOUBLE(Y),8) OR GET$Y(1);
        ZZ=0;
        GET$Z(2)=GET$X(2);
        CALL SET$T;
        IF NOT CARRY THEN DO;
            I=1;
            GOTO ENT2;
            END;
        CALL SHIFT$X;
        GET$Z(2)=(GET$Z(2) AND 7FH)-1;
        I=0;
 ENT1:  CALL SET$T;
        IF CARRY THEN DO;
            CALL SHIFT$X;
            GOTO ENT3;
            END;
   /* NO CARRY, ADD ONE TO DIVIDEND TO MANTISSA AND SHIFT LEFT */
 ENT2:  XX=T;
        ZZ=ZZ+1;
   /* CARRY SHIFT DIVIDEND MANTISSA LEFT */
 ENT3:  ZZ=SHL(ZZ,1);
        I=I+1;
        IF I< 16 THEN GOTO ENT1;
        Z=HIGH(ZZ);
        GET$Z(1)=LOW(ZZ);
    /* SET EXPONENT */
        GET$Z(2)=(((GET$X(2) AND 80H) XOR (GET$Y(2) AND 80H)) OR
                  ((GET$Z(2) AND 7FH) - (GET$Y(2) AND 7FH)));
        RETURN ZA;
END DIV;

    /* FLOATING POINT SQUARE ROOT ROUTINE */
SQRT:  PROCEDURE (XA) ADDRESS;
        DECLARE (XA,YA,YY) ADDRESS,
                X BASED XA BYTE,
                Y BASED YA BYTE,
                I BYTE;
```

```
        GET$Y: PROCEDURE (I) BYTE;
               DECLARE I BYTE;
               RETURN Y(I);
        END GET$Y;

        GET$X: PROCEDURE (I) BYTE;
               DECLARE I BYTE;
               RETURN X(I);
        END GET$X;

        YMAN: PROCEDURE;
              GET$Y(2)=((GET$Y(2) AND 3FH)-1) OR (GET$Y(2) AND 0C0H);
        END YMAN;

        YA=XA;
        GET$Y(2)=SHL(((GET$X(2) AND 3FH)+1),1) OR (GET$X(2) AND 0C0H);
    /* NEWTON'S ITERATION FOR SQUARE ROOT ESTIMATION */
        DO I= 1 TO 3;
            YA=ADD(DIV(XA,YA),YA);
            CALL YMAN;
        END;
        RETURN YA;
END SQRT;

    /* TABLE LOOK-UP PHASE ANGLE ROUTINE - UNWRITTEN */
ATAN2: PROCEDURE (XA,YA) ADDRESS;
       DECLARE (XA,YA,ZA) ADDRESS;
         RETURN ZA;
 END ATAN2;


    /* ROUTINE TO COMPARE TWO FLOATING POINT VARIABLES */
COMPARE: PROCEDURE (XA,YA) BYTE;
     DECLARE X BASED XA BYTE,
             Y BASED YA BYTE,
             (XA,YA,CHECK1,CHECK2) ADDRESS;

     GET$X: PROCEDURE (I) BYTE;
            DECLARE I BYTE;
            RETURN X(I);
     END GET$X;

     GET$Y: PROCEDURE (I) BYTE;
            DECLARE I BYTE;
            RETURN Y(I);
     END GET$Y;

     IF (GET$X(2) AND 80H) = (GET$Y(2) AND 80H) THEN DO;
```

```
  /* EQUAL EXPONENTS */
          IF (GET$X(2) AND 7FH) > (GET$Y(2) AND 7FH) THEN RETURN 0;
          IF (GET$X(2) AND 7FH) < (GET$Y(2) AND 7FH) THEN RETURN 2;
          CHECK1=SHL(DOUBLE(X),8) OR GET$X(1);
          CHECK2=SHL(DOUBLE(Y),8) OR GET$Y(1);
          IF CHECK2 < CHECK1 THEN RETURN 2;
          IF CHECK2 > CHECK1 THEN RETURN 0;
          RETURN 1;
          END;
  /* X > Y */
     IF (GET$X(2) AND 80H) > (GET$Y(2) AND 80H) THEN RETURN 0;
  /* Y > X */
     RETURN 2;
END COMPARE;

  /* TABLE LOOK-UP SINE ROUTINE - UNWRITTEN */
SIN: PROCEDURE (XA) ADDRESS;
        DECLARE (XA,ZA) ADDRESS;
        RETURN ZA;
 END SIN;


  /* TABLE LOOK-UP COSINE ROUTINE - UNWRITTEN */
COS: PROCEDURE (XA) ADDRESS;
        DECLARE (XA,ZA) ADDRESS;
        RETURN ZA;
 END COS;


  /* TABLE LOOK-UP ARC TANGENT ROUTINE UNWRITTEN */
ATAN: PROCEDURE (XA) ADDRESS;
          DECLARE (XA,ZA) ADDRESS;
          RETURN ZA;
END ATAN;


  /* TABLE LOOK-UP ROUTINE: CURRENTLY USED TO FIND DENSITY,SPEED OF SOUND
                            AND THE DRAG COEFFICIENT */
TABLE: PROCEDURE (ENTRY,NUM,LOOK,ENT,VAL,DIFF,MAX) ADDRESS;
      DECLARE (ENTRY,ENT,VAL,DIFF,VALUE,TEMP) ADDRESS,
              BTEMP BASED TEMP BYTE,
              ENTB BASED ENT BYTE,
              VALB BASED VAL BYTE,
              DIFFB BASED DIFF BYTE,
              (NUM,LOOK,MAX,K) BYTE;

      GET$TAB: PROCEDURE (X,I) BYTE;
```

```
            DECLARE (X,I) BYTE;
            RETURN X(I);
        END GET$TAB;

        IF NUM=1 THEN LOOK=LOOK+MAX;
         DO CASE COMPARE(ENTRY,GET$TAB(ENTB,LOOK));
           GOTO TAB5;
           GOTO TAB10;
           GOTO TAB1;
         END;
TAB1:    IF LOOK <= 0 THEN GOTO TAB10;
         LOOK=LOOK-3;
         DO CASE COMPARE(ENTRY,GET$TAB(ENTB,LOOK));
           GOTO TAB4;
           GOTO TAB10;
           GOTO TAB1;
         END;
TAB4:    K=LOOK;
         GOTO TAB9;
TAB5:    IF LOOK >= MAX THEN GOTO TAB10;
         LOOK=LOOK+3;
         DO CASE COMPARE(ENTRY,GET$TAB(ENTB,LOOK));
           GOTO TAB5;
           GOTO TAB10;
           GOTO TAB8;
         END;
TAB8:    K=LOOK-3;
TAB9:    TEMP=GET$TAB(ENTB,K);
         TEMP(2)=GET$TAB(BTEMP,2) XOR 80H;
         VALUE=ADD(GET$TAB(VALB,K),MULT(ADD(ENTRY,TEMP),GET$TAB(DIFF,K)));
         GOTO TAB11;
TAB10:   VALUE=VAL(LOOK);
TAB11:   RETURN VALUE;
END TABLE;


  /* BOMB DROP SIMULATION ROUTINE */
TRAJ: PROCEDURE;
        DECLARE (DTO,VE2B,VE3B,DELT1,DELT2,CAT) ADDRESS,
                DELT ADDRESS,
                (IROCK,JROCK,JSTAGE,NMACH,KGATE) BYTE INITIAL(0,0,0),
                (DT,TEMP,TEMP1,TEMP2,GAMMA,VEL1) ADDRESS,
                MMACH (2) BYTE INITIAL (46,0),
                (LKRHO,LKVSB,LKCD) BYTE,
                (YY1,YY2,YY3,YY4,SY1,SY2,SY3,SY4,ST) ADDRESS,
                BDELT1 BASED DELT1 BYTE,
                BDELT2 BASED DELT2 BYTE,
                BDELT  BASED DELT  BYTE,
```

```
            BTEMP  BASED TEMP  BYTE;
            BTEMP1 BASED TEMP1 BYTE;
            BTEMP2 BASED TEMP2 BYTE;
            BDT    BASED DT    BYTE;

 /* RUNGA KUTTA INTEGRATION ROUTINE */
EINT: PROCEDURE (DT);
      DECLARE (YM11,YM12,YM13,YM14,YM21,YM22,YM23,YM24,
               YM31,YM32,YM33,YM34,YM41,YM42,YM43,YM44,
               YT1,YT2,YT3,YT4,DY1,DY2,DY3,DY4,DT,SIX) ADDRESS;
 /* PROCEDURE TO SOLVE FIRST ORDER DERIVATIVES */
 DERIV: PROCEDURE(YY1,YY2,YY3,YY4);
      DECLARE (YY1,YY2,YY3,YY4,VEL,QS,MACH,DEN,TEMP,GR) ADDRESS,
              BTEMP BASED TEMP BYTE, MM BYTE;
      DY2=YY4;
      DY3=YY1;
      VEL=SQRT(ADD(MULT(DY2,DY2),MULT(DY3,DY3)));
      QS=MULT(TABLE(YY2,0,LKRHO,H,RHOT,DRHO,61),MULT(VEL,BS(JSTAGE)));
      MACH=MULT(VEL,TABLE(YY2,0,LKVSB,H,VSBT,DVSB,61));
      TEMP=QS;
      BTEMP(2)=BTEMP(2) XOR 80H;
      DEN=MULT(TABLE(MACH,JROCK,LKCD,AMACH,ACOEF,DCD,MMACH(JROCK)),
               MULT(TEMP,MASS(JSTAGE)));
      DY1=MULT(DEN,YY1);
      DY4=ADD(MULT(DEN,YY4),GR);
      RETURN;
 END DERIV;

      DIV2: PROCEDURE (XA) ADDRESS;
          DECLARE (XA) ADDRESS, X BASED XA BYTE;
          X(2)=X(2)-1;
          RETURN XA;
      END DIV2;

      MULT2: PROCEDURE (XA) ADDRESS;
          DECLARE (XA) ADDRESS, X BASED XA BYTE;
          X(2)=X(2)+1;
          RETURN XA;
      END MULT2;

     CALL DERIV (YY1,YY2,YY3,YY4);
     YM11=MULT(DT,DY1); YM12=MULT(DT,DY2);
     YM13=MULT(DT,DY3); YM14=MULT(DT,DY4);
     YT1=ADD(YY1,DIV2(YM11)); YT2=ADD(YY2,DIV2(YM12));
     YT3=ADD(YY3,DIV2(YM13)); YT4=ADD(YY4,DIV2(YM14));
     CALL DERIV (YT1,YT2,YT3,YT4);
```

```
      YM21=MULT(DT,DY1); YM22=MULT(DT,DY2);
      YM23=MULT(DT,DY3); YM24=MULT(DT,DY4);
      YT1=ADD(YY1,DIV2(YM21)); YT2=ADD(YY2,DIV2(YM22));
      YT3=ADD(YY3,DIV2(YM23)); YT4=ADD(YY4,DIV2(YM24));
      CALL DERIV (YT1,YT2,YT3,YT4);
      YM31=MULT(DT,DY1); YM32=MULT(DT,DY2);
      YM33=MULT(DT,DY3); YM34=MULT(DT,DY4);
      YT1=ADD(YY1,YM31); YT2=ADD(YY2,YM32);
      YT3=ADD(YY3,YM33); YT4=ADD(YY4,YM34);
      CALL DERIV (YT1,YT2,YT3,YT4);
      YM41=MULT(DT,DY1); YM42=MULT(DT,DY2);
      YM43=MULT(DT,DY3); YM44=MULT(DT,DY4);
      YY1=ADD(YY1,DIV(ADD(ADD(YM11,MULT2(YM21)),ADD(MULT2(YM31),YM41))
                  ,SIX));
      YY2=ADD(YY2,DIV(ADD(ADD(YM12,MULT2(YM22)),ADD(MULT2(YM32),YM42))
                  ,SIX));
      YY3=ADD(YY3,DIV(ADD(ADD(YM13,MULT2(YM23)),ADD(MULT2(YM33),YM43))
                  ,SIX));
      YY4=ADD(YY4,DIV(ADD(ADD(YM14,MULT2(YM24)),ADD(MULT2(YM34),YM44))
                  ,SIX));
      TIME=ADD(TIME,DT);
      RETURN;
END EINT;

     GET$T: PROCEDURE (I) BYTE;
         DECLARE I BYTE;
         RETURN BTEMP(I);
     END GET$T;

     SET$T: PROCEDURE (I);
            DECLARE I BYTE;
            BTEMP(I)=GET$T(I) XOR 80H;
     END SET$T;

      DT=DTO;
      TEMP=ADD(VEL,VE2B);
      GAMMA=ADD(ALPHA,ATAN2(VE3B,TEMP));
      VEL1=SQRT(ADD(MULT(TEMP,TEMP),MULT(VE3B,VE3B)));
      IF BDELT1 <> 0 THEN DO;
         BDELT2(2)=BDELT2(2) XOR 80H;
         DELT=ADD(DELT1,MULT(DELT2,VEL1));
         END;
      YY4=MULT(VEL1,SIN(GAMMA));
      YY1=MULT(VEL1,COS(GAMMA));
      YY2=ZAC;
      YY3=DR;
      SY4=YY4;
      NMACH=MMACH;
```

```
        IF IROCK > 0 THEN GOTO PREDL;
SING:   CALL EINT(DT);
        GO TO CKIMP;
PREDL:  KGATE=1;
        DO CASE COMPARE(DELT,DT);
          GOTO NEX1;
          GOTO NEX1;
          DT=DELT;
        END;
NEX1:   IF BDELT(2) AND 80H <> 0 THEN KGATE=4;
          ELSE IF BDELT=0 THEN KGATE=4;
DUAL:   CALL EINT(DT);
        DO CASE KGATE-1;
CASE1:   DO CASE COMPARE(ADD(TIME,DT),DELT);
           GOTO TCAS2;
           GOTO TCAS1;
           GOTO CKIMP;
         END;
CASE2:   GOTO CASE3;
CASE3:   DO;
           TEMP=TIME;
           CALL SET$T(2);
           TEMP=ADD(DELT,TEMP);
           IF (GET$T(2) AND 7FH) < 118 THEN GOTO TCAS3;
         END;
CASE4:   GOTO CKIMP;
        END;
TCAS1:  KGATE=3;
        GOTO CKIMP;
TCAS2:  KGATE=2;
        TEMP=TIME;
        CALL SET$T(2);
        DT=ADD(DELT,TEMP);
        IF (BDT(2) AND 7FH) > 115 THEN GOTO CKIMP;
TCAS3:  KGATE=4;
        DT=DTO;
        JRCCK=1;
        JSTAGE=3;
        NMACH=MMACH(1);
CKIMP:  TEMP=CAT;
        CALL SET$T(2);
        TEMP=ADD(YY2,TEMP);
        IF (GET$T(2) AND 7FH) < 124 THEN GOTO FINISH;
        DO CASE COMPARE(YY2,CAT);
          GOTO SAVE;
          GOTO FINISH;
          GOTO INTPL;
        END;
```

```
SAVE:   SY1=YY1;
        SY2=YY2;
        SY3=YY3;
        SY4=YY4;
        ST=TIME;
        IF (NMACH=0) OR (IROCK>0) THEN GOTO DUAL;
        GOTO SING;
INTPL:  TEMP=TIME;
        CALL SET$T(2);
        TEMP1=SY2;
        BTEMP1(2)=BTEMP1(2) XOR 80H;
        TEMP2=CAT;
        BTEMP2(2)=BTEMP2(2) XOR 80H;
        DT=DIV(MULT(ADD(SY2,TEMP2),ADD(ST,TEMP)),ADD(YY2,TEMP1));
        TIME=ST;
        YY1=SY1;
        YY2=SY2;
        YY3=SY3;
        YY4=SY4;
        IF (NMACH=0) OR (IROCK>0) THEN GOTO DUAL;
        GOTO SING;
FINISH: DR=YY3;
        RETURN;
END TRAJ;

  /* VELOCITY FIT ROUTINE */
VELFIT: PROCEDURE (K);
      DECLARE (F1SUM1,F2SUM1,F1SUM2,F2SUM2,F1SUM3,F2SUM3,
               V1SUM1,V2SUM1,V1SUM2,V2SUM2,V1SUM3,V2SUM3) ADDRESS,
              (O2,O3,RA0,RA1,RA2,TA0,TA1,TA2,PA0,PA1,PA2) ADDRESS,
              (TEMP,TEMP1,POS1,POS2,POS3,VEL1,VEL2,VEL3) ADDRESS,
              (SO1,SO2,SO3,TWDFN,SXDFN) ADDRESS,
              (K,K1,K2,K4,II) BYTE,
              BO2   BASED O2   BYTE,
              BO3   BASED O3   BYTE,
              BTEMP BASED TEMP BYTE,
             BTEMP1 BASED TEMP1 BYTE,
              BRA1  BASED RA1  BYTE,
              BTA1  BASED TA1 BYTE,
              BPA1  BASED PA1  BYTE;

      CONVERT: PROCEDURE (I) ADDRESS;
          DECLARE I BYTE, X ADDRESS;
          RETURN X;
      END CONVERT;

      GET$X: PROCEDURE (X,I) BYTE;
          DECLARE (X,I) BYTE;
```

```
        RETURN X(I);
    END GET$X;

    INIT: PROCEDURE;
        F1SUM1=ZERO;      F1SUM2=ZERO;      F1SUM3=ZERO;
        F2SUM1=ZERO;      F2SUM2=ZERO;      F2SUM3=ZERO;
        V1SUM1=ZERO;      V1SUM2=ZERO;      V1SUM3=ZERO;
        V2SUM1=ZERO;      V2SUM2=ZERO;      V2SUM3=ZERO;
    END INIT;

/* CALCULATE AO */
    CALL INIT;
    K1=15-K;      K=K*3;      K4=K1*3-3;
    IF K1=0 THEN GOTO ENDPKL;
    DO II=0 TO K4 BY 3;
     K2=K+II+3;
     V1SUM1=ADD(V1SUM1,GET$X(BTRK1,K2));
     V1SUM2=ADD(V1SUM2,GET$X(BTRK2,K2));
    END;
    F1SUM1=MULT(CONVERT(K1),BAR1);
    F1SUM2=MULT(CONVERT(K1),BAR2);
    F1SUM3=MULT(CONVERT(K1),BAR3);
ENDPKL: DO II=0 TO K BY 3;
     V2SUM1=ADD(V2SUM1,GET$X(BTRK1,K2));
     V2SUM2=ADD(V2SUM2,GET$X(BTRK2,K2));
     V2SUM3=ADD(V2SUM3,GET$X(BTRK3,K2));
    END;
    F2SUM1=MULT(CONVERT(15-K1),TILDA1);
    F2SUM2=MULT(CONVERT(15-K1),TILDA2);
    F2SUM3=MULT(CONVERT(15-K1),TILDA3);
    TEMP=V1SUM1;    BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
    TEMP1=V2SUM1;    BTEMP1(2)=GET$X(BTEMP,2) XOR 80H;
    RAO=MULT(ADD(ADD(F1SUM1,F2SUM1),ADD(TEMP,TEMP1)),SO1);
    TEMP=V1SUM2;    BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
    TEMP1=V2SUM2;    BTEMP1(2)=GET$X(BTEMP,2) XOR 80H;
    IAO=MULT(ADD(ADD(F1SUM2,F2SUM2),ADD(TEMP,TEMP1)),SO1);
    TEMP=V1SUM3;    BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
    TEMP1=V2SUM3;    BTEMP1(2)=GET$X(BTEMP,2) XOR 80H;
    PAO=MULT(ADD(ADD(F1SUM3,F2SUM3),ADD(TEMP,TEMP1)),SO1);

/* CALCULATE A1 */
    CALL INIT;
    IF K1=0 THEN GOTO ENDPKL1;
    DO II=0 TO K4 BY 3;
     K2=K+II+3;
     F1SUM1=ADD(F1SUM1,GET$X(BO2,II));
     F1SUM2=ADD(F1SUM2,GET$X(BO2,II));
     F1SUM3=ADD(F1SUM3,GET$X(BO2,II));
```

```
        V1SUM1=ADD(V1SUM1,MULT(GET$X(BC2,II),GET$X(BTRK1,K2)));
        V1SUM2=ADD(V1SUM2,MULT(GET$X(BC2,II),GET$X(BTRK2,K2)));
        V1SUM3=ADD(V1SUM3,MULT(GET$X(BC2,II),GET$X(BTRK3,K2)));
      END;
      F1SUM1=MULT(F1SUM1,BAR1);
      F1SUM2=MULT(F1SUM2,BAR1);
      F1SUM3=MULT(F1SUM3,BAR3);
ENDPKL1: DO II= K4+3 TO 45 BY 3;
        F2SUM1=ADD(F2SUM1,GET$X(BC2,II));
        F2SUM2=ADD(F2SUM2,GET$X(BC2,II));
        F2SUM3=ADD(F2SUM3,GET$X(BC2,II));
      END;
      F2SUM1=MULT(F2SUM1,TILDA1);
      F2SUM2=MULT(F2SUM2,TILDA2);
      F2SUM3=MULT(F2SUM3,TILDA2);
      DO II=0 TO K BY 3;
        K2=K4+II+3;
        V2SUM1=ADD(V2SUM1,MULT(GET$X(BC2,K2),GET$X(BTRK1,II)));
        V2SUM2=ADD(V2SUM2,MULT(GET$X(BC2,K2),GET$X(BTRK2,II)));
        V2SUM3=ADD(V2SUM3,MULT(GET$X(BC2,K2),GET$X(BTRK3,II)));
      END;
      TEMP=V1SUM1;    BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
      TEMP1=V2SUM1;    BTEMP1(2)=GET$X(BTEMP,2) XOR 80H;
      RA1=MULT(ADD(ADD(F1SUM1,F2SUM1),ADD(TEMP,TEMP1)),SO2);
      TEMP=V1SUM2;    BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
      TEMP1=V2SUM2;    BTEMP1(2)=GET$X(BTEMP,2) XOR 80H;
      TA1=MULT(ADD(ADD(F1SUM2,F2SUM2),ADD(TEMP,TEMP1)),SO2);
      TEMP=V1SUM3;    BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
      TEMP1=V2SUM3;    BTEMP1(2)=GET$X(BTEMP,2) XOR 80H;
      PA1=MULT(ADD(ADD(F1SUM3,F2SUM3),ADD(TEMP,TEMP1)),SO2);

  /* CALCULATE A2 */
      CALL INIT;
      IF K1=0 THEN GOTO ENDPKL2;
      DO II=0 TO K4 BY 3;
        K2=K+II+3;
        F1SUM1=ADD(F1SUM1,GET$X(BO3,II));
        F1SUM2=ADD(F1SUM2,GET$X(BO3,II));
        F1SUM3=ADD(F1SUM3,GET$X(BO3,II));
        V1SUM1=ADD(V1SUM1,MULT(GET$X(BO3,II),GET$X(BTRK1,K2)));
        V1SUM2=ADD(V1SUM2,MULT(GET$X(BO3,II),GET$X(BTRK2,K2)));
        V1SUM3=ADD(V1SUM3,MULT(GET$X(BO3,II),GET$X(BTRK3,K2)));
      END;
      F1SUM1=MULT(F1SUM1,BAR1);
      F1SUM2=MULT(F1SUM2,BAR2);
      F1SUM3=MULT(F1SUM3,BAR3);
ENDPKL2: DO II= K4+3 TO 45 BY 3;
        F2SUM1=ADD(F2SUM1,GET$X(BO3,II));
```

```
        F2SUM2=ADD(F2SUM2,GET$X(BO3,II));
        F2SUM3=ADD(F2SUM3,GET$X(BO3,II));
      END;
      F2SUM1=MULT(F2SUM1,TILDA1);
      F2SUM2=MULT(F2SUM2,TILDA2);
      F2SUM3=MULT(F2SUM3,TILDA3);
      DO II=0 TO K BY 3;
        K2=K4+II+3;
        V2SUM1=ADD(V2SUM1,MULT(GET$X(BO3,K2),GET$X(BTRK1,II)));
        V2SUM2=ADD(V2SUM2,MULT(GET$X(BO3,K2),GET$X(BTRK2,II)));
        V2SUM3=ADD(V2SUM3,MULT(GET$X(BO3,K2),GET$X(BTRK3,II)));
      END;
      TEMP=V1SUM1;    BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
      TEMP1=V2SUM1;   BTEMP1(2)=GET$X(BTEMP,2) XOR 80H;
      RA2=MULT(ADD(ADD(F1SUM1,F2SUM1),ADD(TEMP,TEMP1)),SO3);
      TEMP=V1SUM2;    BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
      TEMP1=V2SUM2;   BTEMP1(2)=GET$X(BTEMP,2) XOR 80H;
      TA2=MULT(ADD(ADD(F1SUM2,F2SUM2),ADD(TEMP,TEMP1)),SO3);
      TEMP=V1SUM3;    BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
      TEMP1=V2SUM3;   BTEMP1(2)=GET$X(BTEMP,2) XOR 80H;
      PA2=MULT(ADD(ADD(F1SUM3,F2SUM3),ADD(TEMP,TEMP1)),SO3);

   /* COMPUTE POLAR POSITIONS AND VELOCITES */
      BRA1(2)=GET$X(BRA1,2) XOR 80H;
      BTA1(2)=GET$X(BTA1,2) XOR 80H;
      BPA1(2)=GET$X(BPA1,2) XOR 80H;
      POS1=ADD(ADD(RA0,RA1),RA2);
      POS2=ADD(ADD(TA0,TA1),TA2);
      POS3=ADD(ADD(PA0,PA1),PA2);
      VEL1=ADD(MULT(TWOFN,RA1),MULT(SXDFN,RA2));
      VEL2=ADD(MULT(TWOFN,TA1),MULT(SXDFN,TA2));
      VEL3=ADD(MULT(TWOFN,PA1),MULT(SXDFN,PA2));

   /* CALCULATE CARTESIAN POSITIONS AND VELOCITIES */
      TEMP=POS1;      BTEMP(2)=GET$X(BTEMP,2) XOR 80H;
      XDOT=MULT(MULT(SIN(POS2),COS(POS3)),VEL1);
      XDOT=ADD(XDOT,MULT(MULT(POS1,COS(POS2)),MULT(COS(POS3),VEL2)));
      XDOT=ADD(XDOT,MULT(MULT(TEMP,SIN(POS2)),MULT(SIN(POS3),VEL3)));
      YDOT=MULT(MULT(COS(POS2),COS(POS3)),VEL1);
      YDOT=ADD(YDOT,MULT(MULT(TEMP,SIN(POS2)),MULT(COS(POS3),VEL2)));
      YDOT=ADD(YDOT,MULT(MULT(TEMP,COS(POS2)),MULT(SIN(POS3),VEL3)));
      ZDOT=ADD(MULT(SIN(POS3),VEL1),MULT(MULT(POS1,COS(POS3)),VEL3));
      XAC=ADD(MULT(MULT(POS1,SIN(POS2)),COS(POS3)),XVAN);
      YAC=ADD(MULT(MULT(POS1,COS(POS2)),COS(POS3)),YVAN);
      ZAC=ADD(MULT(POS1,SIN(POS3)),ZVAN);
      RETURN;
END VELFIT;
```

```
/* MAIN PROGRAM */
SET$AVG:PROCEDURE;
        BAR1=TILDA1;          TILDA1=TRK1;
        BAR2=TILDA2;          TILDA2=TRK2;
        BAR3=TILDA3;          TILDA3=TRK3;
END SET$AVG;

INSERT:PROCEDURE (I);
        DECLARE I BYTE;
        BTRK1(I)=TILDA1-BTRK1(I);
        BTRK2(I)=TILDA2-BTRK2(I);
        BTRK3(I)=TILDA3-BTRK3(I);
END INSERT;

CHANGE$SIGN: PROCEDURE (X);
        DECLARE XA ADDRESS,
                X BASED XA BYTE;
        X(2)=X(2) XOR 80H;
        RETURN;
 END CHANGE$SIGN;


/* PHASE 1 */

/* READ IN VALUES OF R,THETA,PHI INTO TRK1,TRK2,TRK3..*/
        BAR1=TRK1;  BAR2=TRK2;  BAR3=TRK3;
        TILDA1=TRK1;  TILDA2=TRK2;  TILDA3=TRK3;
INST:   CALL INSERT (I);
        IF PICKL=1 THEN GOTO PICKLE;
        I=I+3;
        I=I MOD 48;
/* READ IN VALUES AS BEFORE*/
        IF I<>0 THEN GOTO INST;
        CALL SET$AVG;
        GOTO INST;

/* PHASE 2 */

PICKLE:     CALL VELFIT(I/3);
        XDOT=MULT(XDOT,T);
        YDOT=MULT(YDOT,T);
        ZDOT=MULT(ZDOT,T);
        GAMMA1=ATAN(DIV(XDOT,YDOT));
        XDOT=ADD(XDOT,XWIND);
        YDOT=ADD(YDOT,YWIND);
        XX=ADD(MULT(XDOT,XDOT),MULT(YDOT,YDOT));
        VEL=SQRT(ADD(XX,MULT(ZDOT,ZDOT)));
```

```
            ALPHA=ATAN(DIV(ZDOT,SQRT(XX)));
   /* PHASE 3 */
            CALL TRAJ;
   /* PHASE 4 */
            GAMMA=ATAN(DIV(XDOT,YDOT));
            CALL CHANGE$SIGN(XAC);
            CALL CHANGE$SIGN(XWIND);
            CALL CHANGE$SIGN(YAC);
            CALL CHANGE$SIGN(YWIND);
            IMPX=ADD(ADD(MULT(DR,SIN(GAMMA)),XAC),MULT(XWIND,TIME));
            IMPY=ADD(ADD(MULT(DR,COS(GAMMA)),YAC),MULT(YWIND,TIME));
            TEMP=IMPY;
            CALL CHANGE$SIGN(TEMP)
            IMPXAC=ADD(MULT(IMPX,COS(GAMMA1)),MULT(TEMP,SIN(GAMMA1)));
            IMPYAC=ADD(MULT(IMPX,SIN(GAMMA1)),MULT(IMPY,COS(GAMMA1)));
            ALPHA=MULT(ALPHA,DEG);
            VEL=DIV(VEL,FPS);
   /*OUTPUT THE APPROPRIATE VALUES*/
EOF
//PASS2 EXEC PGM=PLM2,REGION=100K,COND=(0,NE)
//STEPLIB DD UNIT=2314,VOL=SER=LINDA,DISP=SHR,DSN=C0012.MCS
//FT06F001 DD DUMMY
//PRINTER DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=726)
//INPUT6 DD DSN=*.PASS1.OUTPUT6,DISP=(OLD,PASS)
//INPUT7 DD DSN=*.PASS1.OUTPUT7,DISP=(OLD,PASS)
//OUTPUT7 DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=726)
//SYSUDUMP  DD  SYSOUT=A
//CARD DD *
```

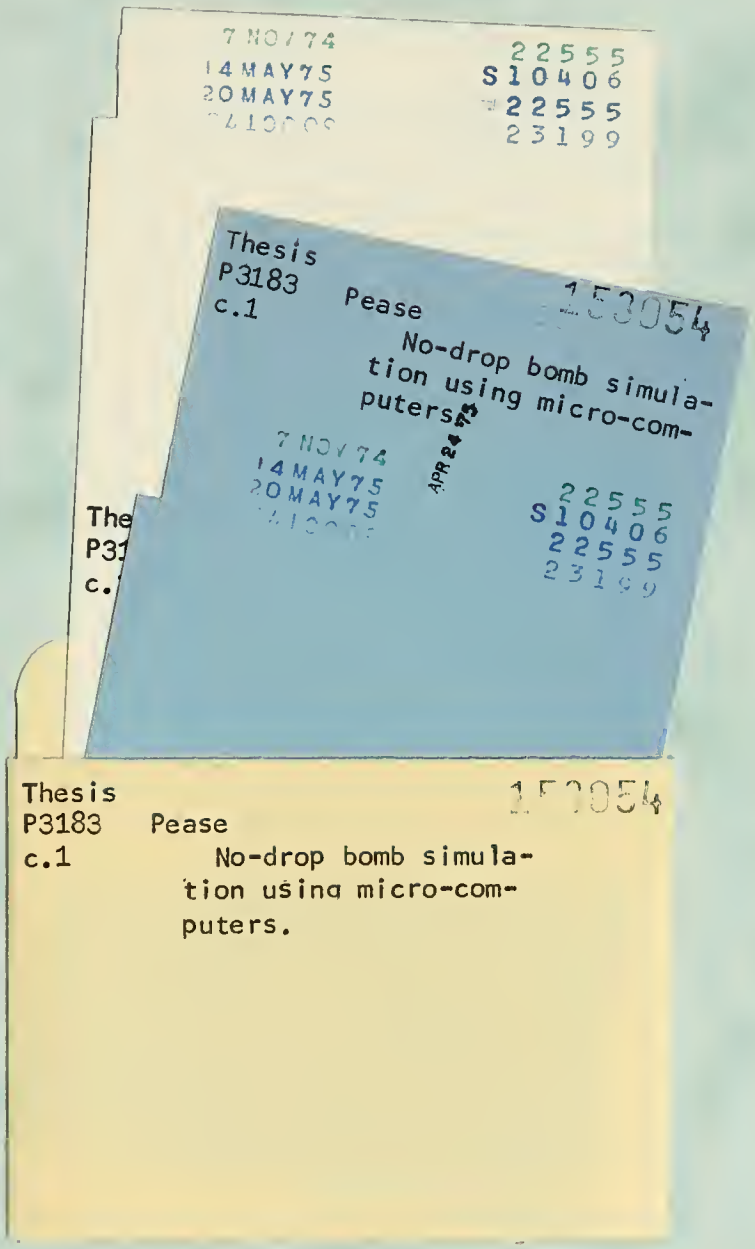
LIST OF REFERENCES

1. McCalla, Thomas Richard, Introduction To Numerical Methods and FORTRAN Programming, John Wiley & Son Inc., New York, London, Sydney, 1967, p. 250-254.

2. Thomson, F. L., Simulation Algorithms (informal).

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 72 Department of Computer Science Naval Postgraduate School Monterey, California 93940	2
4.	Assoc. Professor U. R. Kodres, Code 72Kr Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	Ens. A. J. Pease, USN 4121 N. 33rd. Rd. Arlington, Virginia 22207	1
6.	Jerry Gore Operational Test and Evaluation Force COMOPTEVFOR Norfolk, Virginia 23511	1
7.	Lt. A. A. Pease Lawerence Livermore Laboratory Livermore, California 94550	1
8.	Lee Thomson Commander Naval Weapons Center Code 4074 China Lake, California 93554 (Attn. L. Thomson)	1
9.	Frank Buffen Commander Naval Weapons Center Code 1203 China Lake, California 93554 (Attn. F. Buffen)	1



Thesis
P3183 Pease
c.1 No-drop bomb simulation using micro-computers.